IRobot System User's Manual

IRobotSoft.com

Jan 1, 2015

Catalog

CATALOG 1
1. INTRODUCTION TO THE IROBOT SYSTEM 4
1.1. About the IRobot Manual 4
1.2. What is IRobot 4
1.3. Why Choosing IRobot 5
1.4. Definitions 5
1.5. IRobot Installation 6
1.6. The IRobot System Interface 7
1.7. Robot Execution 8
2. CREATE NEW ROBOTS 9
2.1. Record Actions Automatically 10
2.2. Design Actions Manually 14
A) Go to URL 15
B) A Click 15
C) A List of Links 16
D) Take Data 16
E) Take Table 16
F) Submit a Form 16
G) Logon Form 17
H) Open a Frame 17
I) Sent Emails 17
J) A Schedule 17
2.3. Repeat on Next Pages 18
A) Design Repeat on Next Pages 18
B) Try with Other Repeat Options 19
2.4. Open Links in New Windows 20
2.5. Create Variables 21
A) Create Variables for Extraction Data 21
B) Create Variables for Table Data 23
2.6. Save Variables into Databases 25
2.7. Draw Data from Databases 28
A) Define New Data Sources 29

B) Connect Robot Input with Data Sources 30
2.8. Scheduled Run Tasks *32*
3. BUG FIXING 32
3.1. Debugging Robot Errors *34*
A) Test a Single Action 34
B) Test Run Selected Actions 34
C) View the Log File 34
3.2. Repairing Robot Actions *35*
A) Change the Extraction Query by Wizard 35
B) Check the Action Property 37
3.3. Change HTQL Manually *37*
A) Practice HTQL 38
B) Compose Table Query 40
C) Make It More Robust 42
4. MANAGING MULTIPLE ROBOT TASKS 43
4.1. Calling Robot Tasks *43*
4.2. Organizing Robot Tasks *44*
4.3. Parallelize Robot Tasks *45*
5. IROBOT REFERENCES 45
5.1. Event Definitions *45*
5.2. Internal Variables *47*
5.3. Dataset Variables and Functions *48*
A) Dataset Variables 48
B) Dataset Functions 49
C) Use Dataset Variables for Form Submission 51
5.4. Internal Functions *52*
A) String Functions 52
B) Date Time Functions 53
C) Floating Number Functions 54
D) Data Type Functions 54
E) File Functions 55
F) Interface Functions 56
G) Browser Functions 57
H) Automation Functions 58
I) Browser Proxy Functions 59
J) Crawling Functions 61
K) Setting Functions 62
5.5. Scripting *62*
5.6. Command Line Parameters *64*
5.7. Detailed Action Properties *64*
1) Go to URL 64
2) A Click 65
3) A List of Links 66
4) Take Data 68

5) Take Table 68
6) Submit a Form 69
7) Logon Form 70
8) Open a Frame 70
9) Sent Email 71
10) A Schedule 71
6. FREQUENTLY ASKED QUESTIONS 72
6.1. Can I create a dummy action as a placeholder? *72*
6.2. The robot skips certain actions during navigation! *72*
6.3. How can I scrape data from pop-up windows? *72*
6.4. Save Variables does not work with MS-Access database, but it works with XML *73*
7. ADDITIONAL INFORMATION 73
COPYRIGHT 73

1. Introduction to the IRobot System

1.1. About the IRobot Manual

This manual introduces the use of IRobot system for visual Web scraping and Web automation. The manual is intended for both novice user and experienced users. New users are encouraged to watch **video demos** at: http://irobotsoft.com/help/ before reading this manual.

Readers without any programming experience can still read the first two sections and be able to record robots to automate simple websites. You can also draw data from databases and save data to text files using the visual tools.

With a little programming effort, you can make your robots more reliable. You may improve the query generated automatically by the system, in the form of Hyper-Text Query Language (HTQL), and make them more robust on new pages; you may design robot actions manually to get better control; or use datasets to batch-process data at hand. HTQL specific help can be found from http://htql.net/ (http://htql.net/htql-manual.pdf). Getting familiar with HTQL is useful because IRobot is based on HTQL.

After getting more familiar with the IRobot system, you may start encountering problems when designing more complex robots. Please refer Sections 3 and 4 for tools to locate robot debugs and fix errors. A frequent issue our users reported is the handling of Next pages. You can check out Section 2.3 (Page 18, Repeat on Next Pages) for various options to fix the problem. After reading these sections, readers would be able to design robots for complex Web sites.

The power of IRobot system comes with a full range of advanced functions and utilities that allow you to design the most powerful robots. Section 5 and 6 introduce advanced features like events, datasets, internal functions, task calls, parallelization, scripting, etc. for advanced robot design. They are written in a reference style, so readers can quickly locate relevant parts. With these techniques, you can turn your robot into a powerful and intelligent data machine.

1.2. What is IRobot

IRobot (named for *Internet Robot*) is a visual automation tool to create robot agents, or *irobots*, for Web data aggregation. An irobot agent is able to navigate Web sites, fill in Web forms, extract data, compute and transform data on the fly, and integrate directly with local databases. Using the user-friendly interfaces, you don't need to have programming skills to

create irobots; but with some programming skills, you can create more powerful irobots.

IRobot runs on MicroSoft Windows NT, XP, Vista, and Win 7, and requires MicroSoft Internet Explorer (IE) 6.0 and above for Web automation. Currently IRobot only support IE for Web automation and does not support other Web browsers.

1.3. Why Choosing IRobot

You choose IRobot because:

- You need a visual tool to automate Web form submission and data extraction;
- You want to repeatedly collect data from multiple Web sites;
- You want an easy way to save Web data into databases;
- You want to test your Web sites automatically and repeatedly;
- You want to visually see how data is extracted from the Web;
- You want to compute Web data on the fly.

IRobot is developed entirely in C++ and visual C++, and works extremely efficient. Internally, IRobot embeds an IE Web browser. Running IRobot for Web navigation works exactly like a normal user navigating on the Web via IE. In fact, if you change any IE security/option settings, it will affect IRobot as well.

1.4. Definitions

Robot (or irobot): An encoded script file containing Web navigation and data computation rules. A robot file usually has a suffix ".irb". Robot files are password protected so that personal information for Web submission cannot be read directly by others.

Robot action: An action completes certain Web function, such as Web navigation, form submission and data extraction. Variables and event rules can be defined with a robot action for data computation.

Internal variable: Internal variables maintained by the IRobot system during execution.

User-defined variable: Variables defined by users in the robot. A user-defined variable has a global scope and exists during the entire robot execution. **Variable name** should start with an English letter, and followed by a number of letters, digits, or underscore (_) without any space or special characters. So Abc_1 is a valid variable name, but Abc-1 is not.

Expressions: IRobot follows most standard SQL syntax for **conditions** and **expressions**. A string is enclosed by two single quotation marks ('), such as a 'test string'. IRobot does not recognize double quotation marks ("), so do not use it. If the string itself contains single quotation marks, use another single quot to escape each. So 'It''s a robot' is a valid string.

Within a string, \r, \n, \t, and \\ stand for line-feed, new-line, TAB, and backslash-char, respectively. More supported functions are defined in section 5.4.

Events: Events are defined with robot actions. Events are usually used for **data computation**.

Tuple: Data extracted from a Web page are always treated as a **table**. (Even a single item is considered a table with one row and one column) Each row of the table data is termed a **tuple**. IRobot accesses each tuple (or row) of the table at a time, and do computations in between via **events**.

Robot task: A sequences of robot actions comprises a robot task. A robot task works like a **function** or a procedure and can be called from other tasks via events.

IRobot system: A visual interface for users to design **irobots**.

IRobot Engine: A core library for robot execution written in C++. The IRobot engine can be embedded in higher-layer applications using ActiveX controls. Refer the IRobot ActiveX Programmer's Guide for more help.

HTQL: A Web query language used in IRobot for Web data extraction. Refer The Hyper-Text Query Language (HTQL) – A Web Programmer's Guide from http://htql.net/ for detailed syntax of HTQL. IRobot supports all HTQL syntax in the Guide. IRobot users can practice HTQL following instructions in page 37.

Data source: Any **external database** (text file, XML file, Access DB, MSSQL, MySQL etc.) defined in a robot for batched data processing.

Dataset: An **in-memory database** to store data read from data sources. A dataset is a named object with multiple fields. Each field can be accessed by the dot ('.') operation following the dataset name.

Script: Code in a C-style language supported by IRobot for data computation. Script variables are global during the entire run.

1.5. IRobot Installation

IRobot is published in free evaluation versions in a RAR or ZIP package (irobot-eval.rar or irobot-eval.zip). For installation, download the latest package, create an empty directory, for example, "C:\Program Files\irobot\", herein called IROBOT directory, and exact the "irobot.exe" from the RAR or ZIP package into that directory. The package may contain some ".irb" files for demonstration purpose. Extract those ".irb" files to the same IROBOT directory.

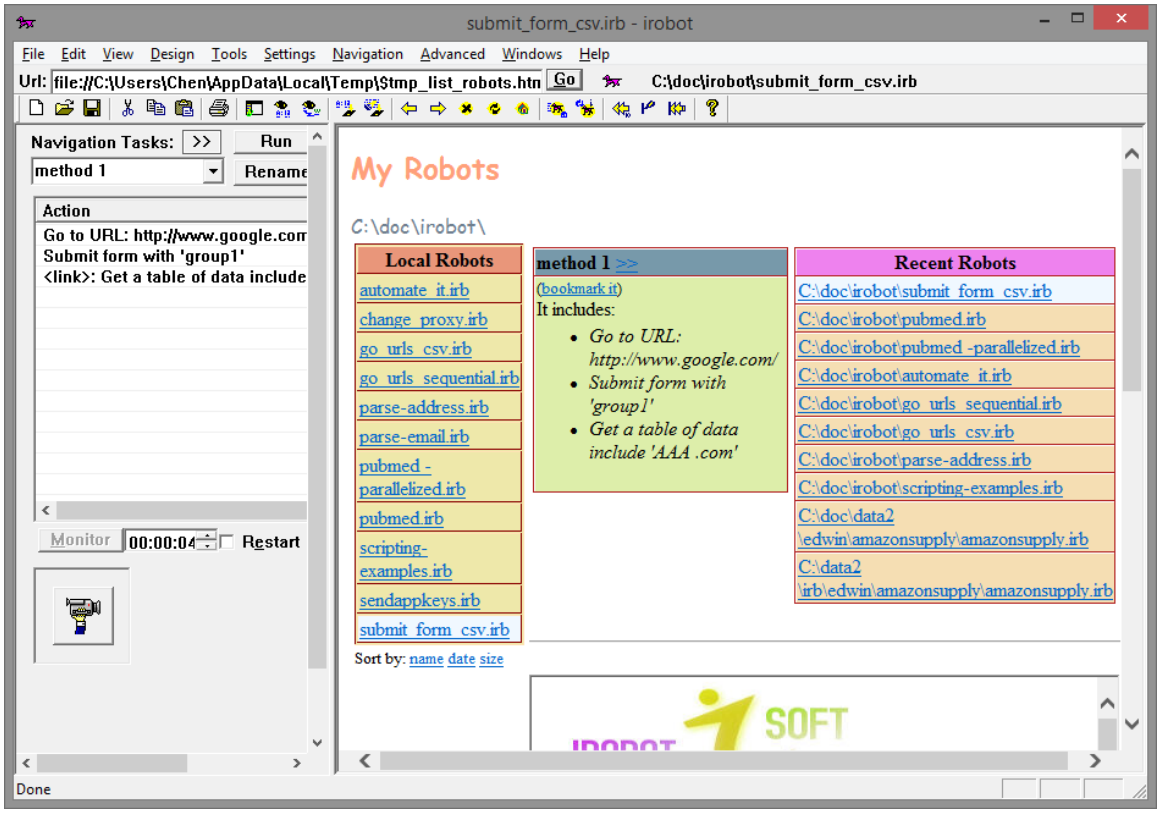
A first run of IRobot will create two additional directories under the IROBOT directory, namely:

IROBOT\data: The default directory for IRobot to save downloaded data.

IROBOT\system: The system directory for IRobot to save temporary data and settings.

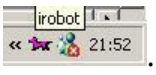
1.6. The IRobot System Interface

The IRobot system interface is shown like:



On the left of the interface, it is a control panel. The control panel displays the content of a robot, and includes buttons for running, recording and renaming robot tasks.

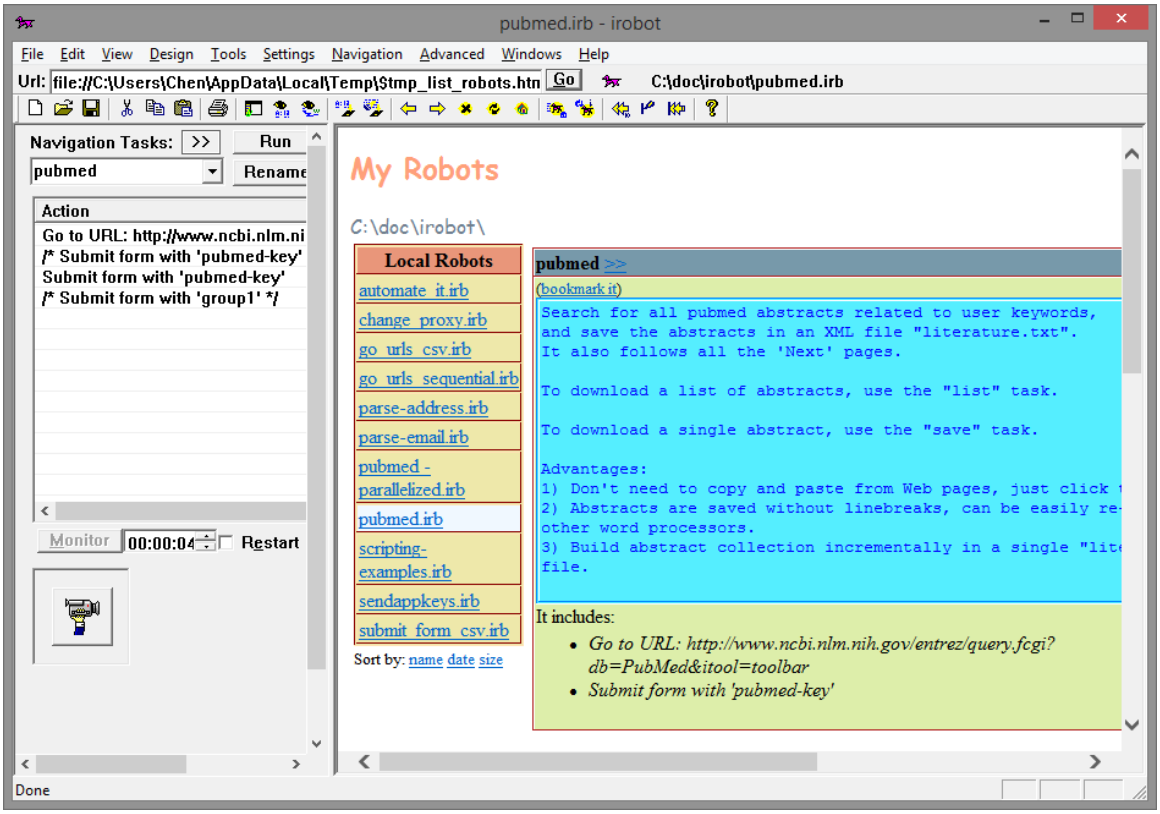
On the right of the interface, there is a browser window showing lists of robots in the IROBOT directory, in the recent execution history, and on the IRobotSoft Web site. You can always show the list of robots by clicking the icon on the toolbar:

There is an icon in the tray-bar shown like 

. If you minimize the IRobot system, IRobot system will be hidden in the tray-bar. Double click the tray-bar to show the IRobot again.

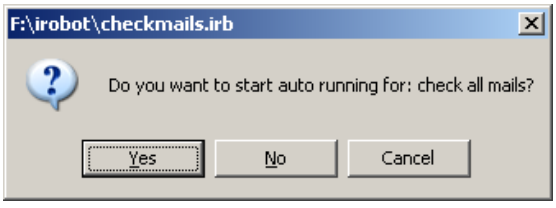
1.7. Robot Execution

Click on a robot file from the right of the IRobot interface to open the robot file, for example, click on the "pubmed.irb" under the "My Robots". The robot tasks and actions will be shown in the left panel. The following figure shows the list of actions in the "pubmed.irb".



Choose a correct robot task from the task drop-down list, such as the "pubmed" task, and click the [Run] button to execute the robot.

Some robots are customized to run automatically once it is open. In this case, IRobot system will show a dialog box to ask whether to run a default robot task, for example, if you open the "checkmails.irb", it will ask you whether to run a default task:



You can click [Yes] to run it or [No] or [Cancel] to stop it.

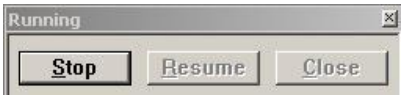
When a robot is running, the tray-bar icon is animated like

.

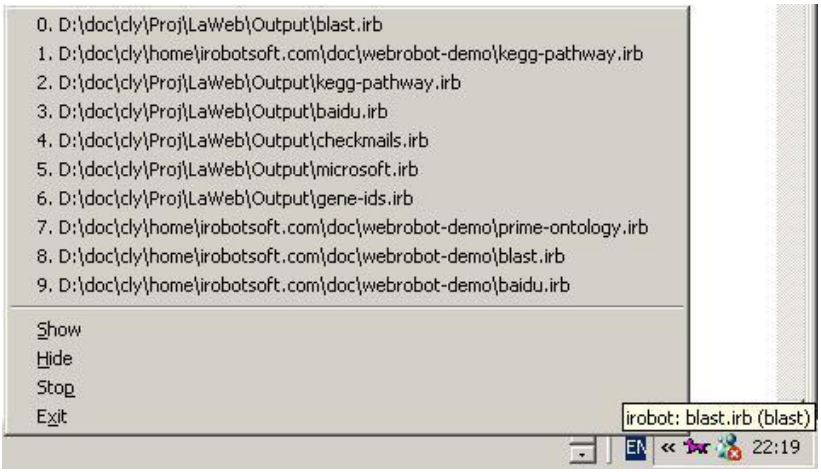
A running robot can be in an active status or in a sleeping status. When a robot is navigating

the Web or processing data, it is in the active status. When a robot is running but is idling, it is in a sleeping status. A sleeping robot is still running but is waiting for the next scheduled time to act on certain thing.

A running robot can be stopped in two ways: one by clicking the [Stop] button on the control bar like



and the other by right-clicking on the irobot tray-bar icon and clicking on the stop menu item, like:



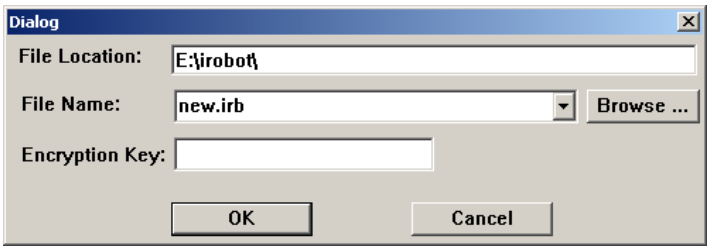
2. Create New Robots

This section demonstrates how to create a new robot to automatically download abstracts from the Pubmed literature search and save the abstracts in a local XML file. This is a very common Web scraping scenario and it is recommended for every new starter to read through. After reading this section and recreating the pubmed robot, you will be able to create similar robots for Web scraping or Web automation.

What is the pubmed robot?
The robot will go to the Pubmed literature search website, ask the user for a keyword. It will then submit the keyword, open a list of links in the page, extract multiple attributes including the abstract from the result page, and follow the "Next" pages continuously. The robot will save the extracted data in an XML file. The final robot can be found from the "pubmed.irb" on the irobotsoft website: http://irobotsoft.com/robots/pubmed.irb. A video demo can be found at: http://irobotsoft.com/help/.

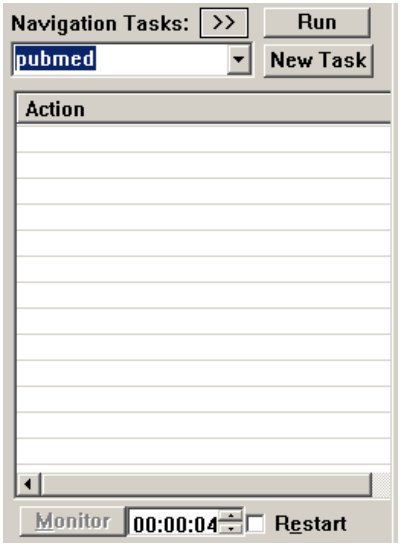
Getting start:
First, we need to create a new robot. Click on the menu File→ New, and create a new robot file. You may want to name the new robot in a suffix of '.irb', for example, 'new.irb', as in

the following interface.



You can set an encryption key for the new robot file. If set, the encryption key will be asked whenever the robot file is open. We will leave it empty for now, and click [OK].

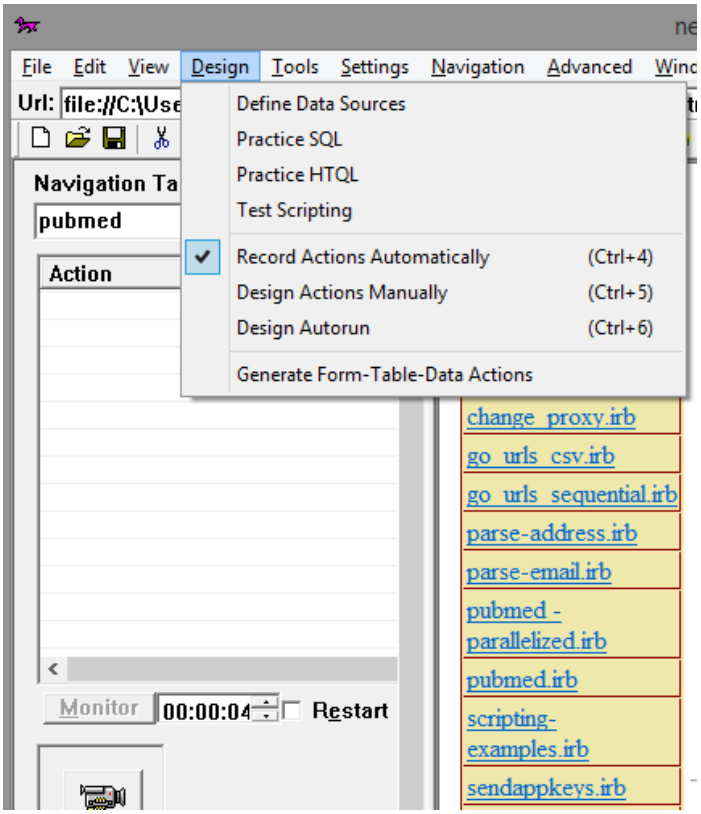
The newly created robot has no any Web navigation task. To create a new navigation task, put a task name in the dropdown box under the Navigation Task Name from the left panel, such as "pubmed", as shown below, and click the [New Task] button to create the new task.



Web actions can be added to the robot task in either of the two ways: 1) by recording; and 2) by step-by-step customization. We will first show how to record a basic "skeleton" of the new robot.

2.1. Record Actions Automatically

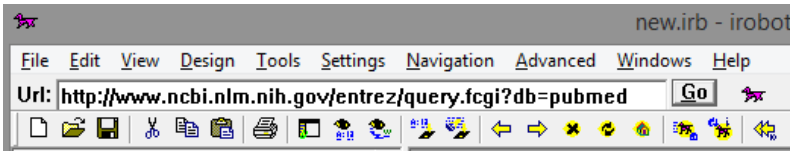
To record Web navigation actions, open the robot recorder from menu: View→ Record Actions Automatically and the robot recorder will be shown in the left panel, as in the following figure:



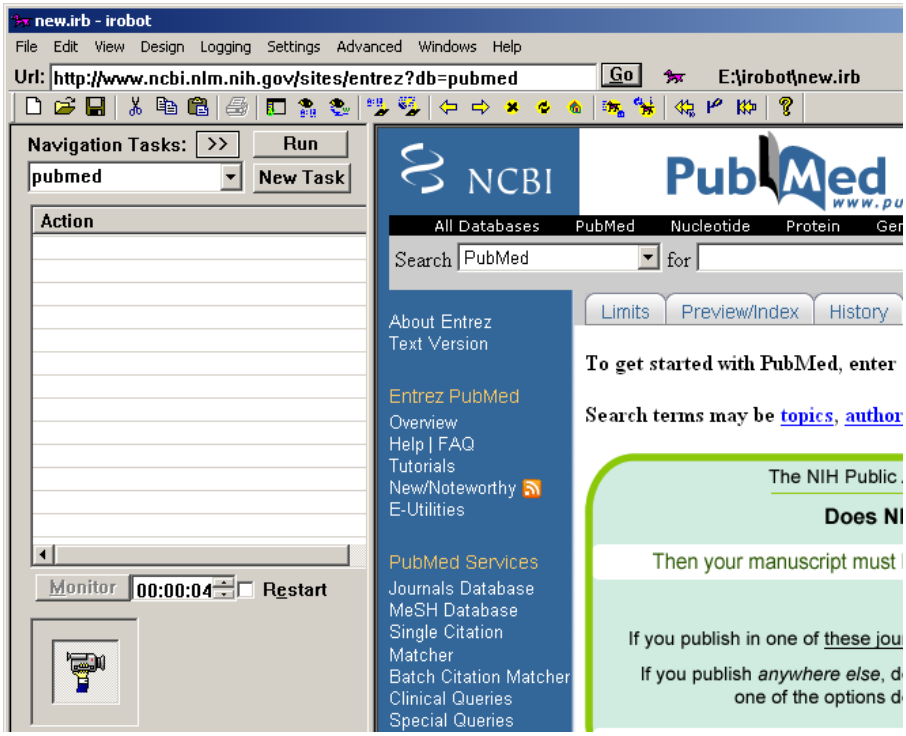
We will press on the recorder button and start recording, like

then navigate from the browser as usual:

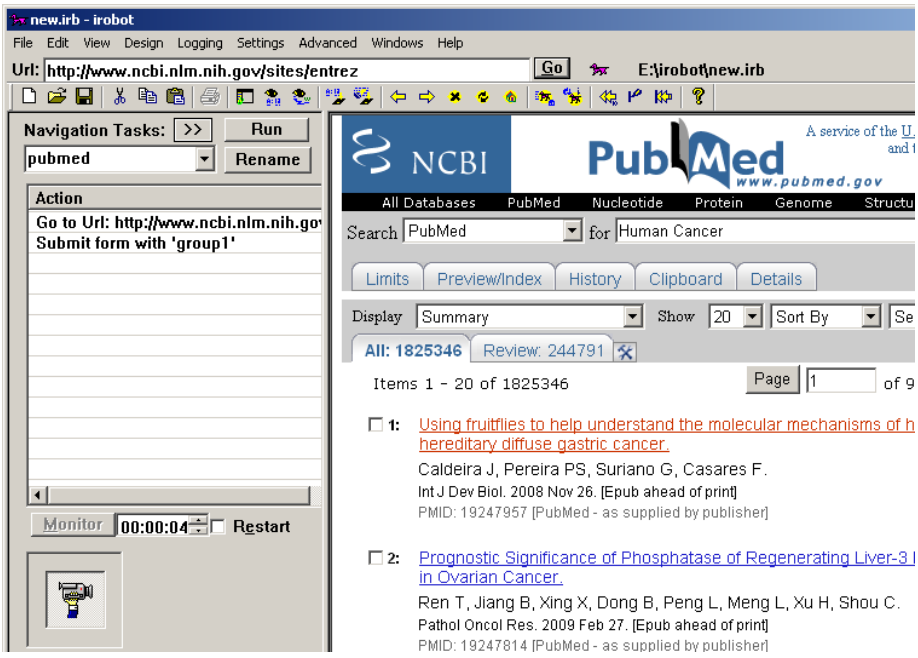
1) Go to the Pubmed web site: Enter "http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=pubmed" in the URL bar and click [Go] as in the following figure:


new.irb - irobot
File Edit View Design Tools Settings Navigation Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=pubmed
Go

The browser will bring up the Pubmed Web site like the following (the Go to URL action will show up in the next step).

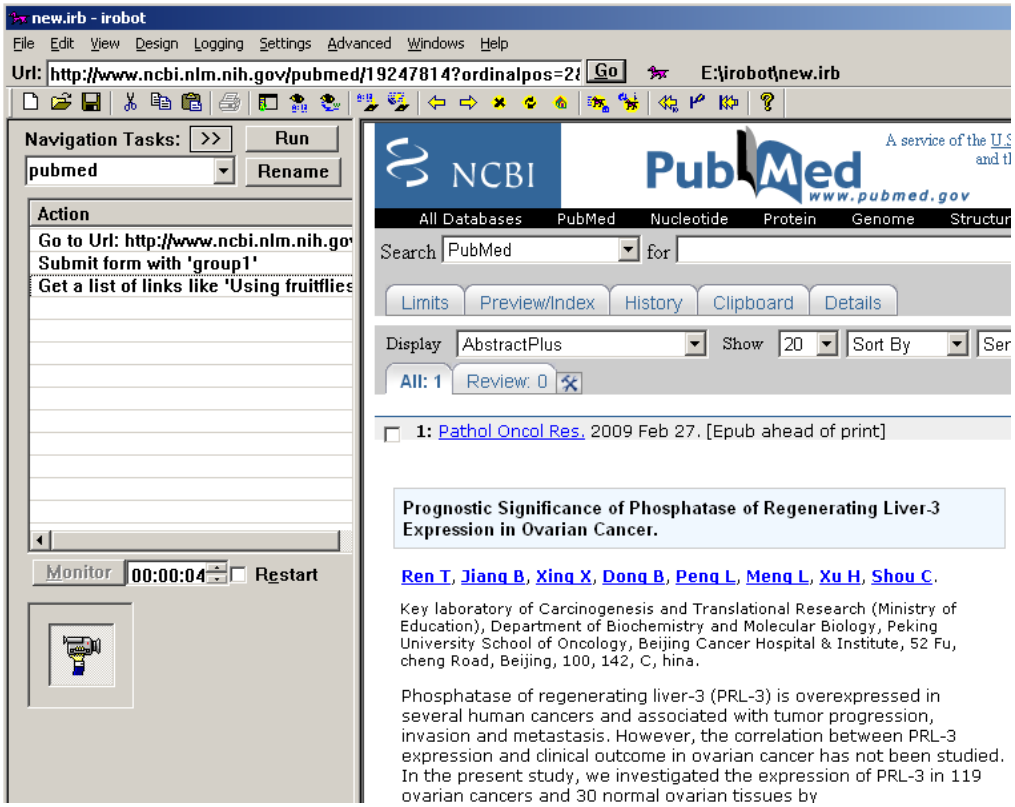


2) Enter a keyword in the Web page, like "Human cancer" and click [Go] in the Web page. It shows:



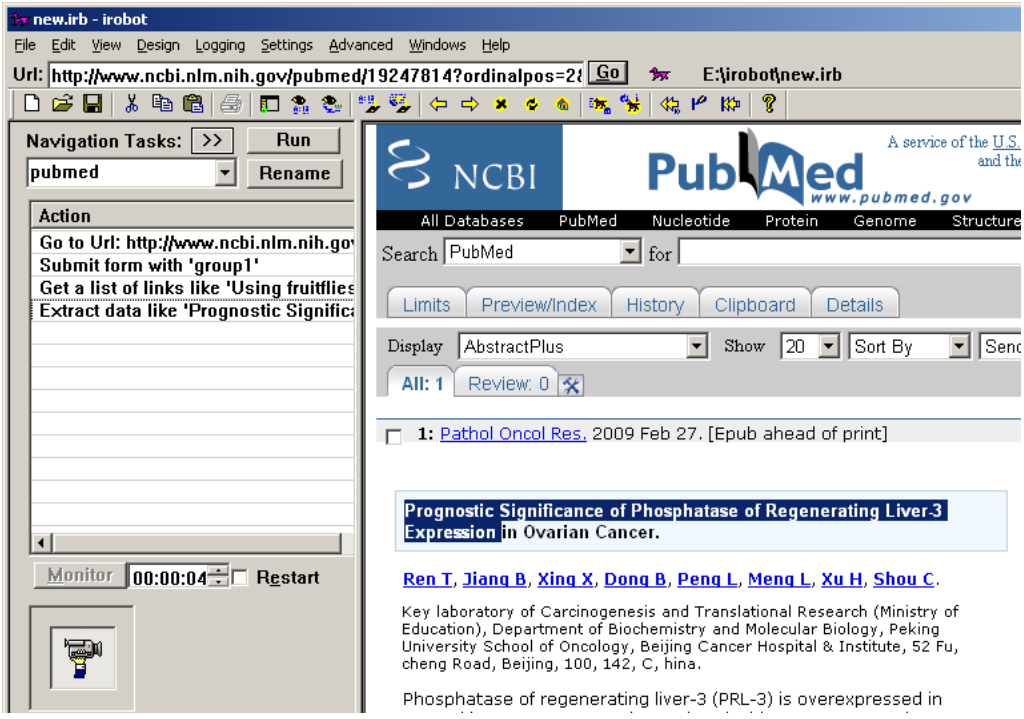
You will notice there is a new task action "Go to Url: ***" followed by another action "Submit form with 'group2'" shown under the action list. (The Go to URL action always appears when the next action is recognized).

3) Click on a link on the result page, like the second link: "Prognostic Significance of ***", and it will show a page like:


new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/pubmed/19247814?ordinalpos=2
Go
E:\irobot\new.irb
Navigation Tasks: >>
Run
pubmed
Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
Get a list of links like 'Using fruitflies
Monitor
00:00:04
Restart
NCBI
PubMed
www.pubmed.gov
All Databases PubMed Nucleotide Protein Genome Structur
Search PubMed for
Limits Preview/Index History Clipboard Details
Display AbstractPlus Show 20 Sort By
All: 1 Review: 0
1: Pathol Oncol Res. 2009 Feb 27. [Epub ahead of print]
Prognostic Significance of Phosphatase of Regenerating Liver-3 Expression in Ovarian Cancer.
Ren T, Jiang B, Xing X, Dong B, Peng L, Meng L, Xu H, Shou C.
Key laboratory of Carcinogenesis and Translational Research (Ministry of Education), Department of Biochemistry and Molecular Biology, Peking University School of Oncology, Beijing Cancer Hospital & Institute, 52 Fu, cheng Road, Beijing, 100, 142, C, hina.
Phosphatase of regenerating liver-3 (PRL-3) is overexpressed in several human cancers and associated with tumor progression, invasion and metastasis. However, the correlation between PRL-3 expression and clinical outcome in ovarian cancer has not been studied. In the present study, we investigated the expression of PRL-3 in 119 ovarian cancers and 30 normal ovarian tissues by

A new action will also appear under the action list: "Get a list of links like ***" (it may take a while before the recorder figures this action out ;). So, be patient, don't try to click too fast when recording robots).

4) Mark the title of the abstract until a new action "Extract data like ***" appears under the action list, like:

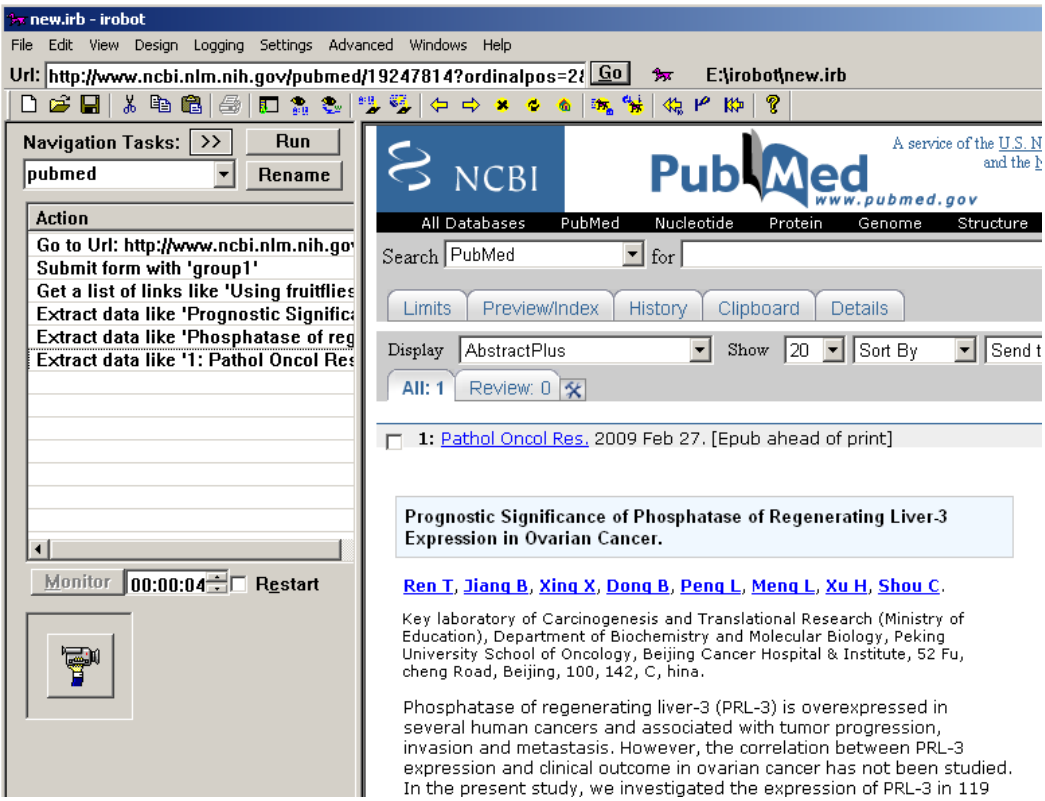

new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/pubmed/19247814?ordinalpos=2
Go
E:\irobot\new.irb
Navigation Tasks: >>
Run
pubmed
Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
Get a list of links like 'Using fruitflies
Extract data like 'Prognostic Significa
Monitor
00:00:04
Restart
NCBI
PubMed
www.pubmed.gov
All Databases PubMed Nucleotide Protein Genome Structure
Search PubMed for
Limits Preview/Index History Clipboard Details
Display AbstractPlus Show 20 Sort By
All: 1 Review: 0
1: Pathol Oncol Res. 2009 Feb 27. [Epub ahead of print]
Prognostic Significance of Phosphatase of Regenerating Liver-3 Expression in Ovarian Cancer.
Ren T, Jiang B, Xing X, Dong B, Peng L, Meng L, Xu H, Shou C.
Key laboratory of Carcinogenesis and Translational Research (Ministry of Education), Department of Biochemistry and Molecular Biology, Peking University School of Oncology, Beijing Cancer Hospital & Institute, 52 Fu, cheng Road, Beijing, 100, 142, C, hina.
Phosphatase of regenerating liver-3 (PRL-3) is overexpressed in

5) Mark the abstract until a new action "Extract data like ***" appears under the action list.

6) Mark the journal information until a new action "Extract data like ***" appears under the action list.

7) Release the recorder button on the left panel to stop recording.

Now we have the skeleton of the "pubmed" robot available as in the following:


new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/pubmed/19247814?ordinalpos=2
Go
E:\irobot\new.irb
Navigation Tasks:
Run
pubmed
Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
Get a list of links like 'Using fruitflies
Extract data like 'Prognostic Significa
Extract data like 'Phosphatase of reg
Extract data like '1: Pathol Oncol Res
Monitor
00:00:04
Restart
NCBI
PubMed
www.pubmed.gov
All Databases PubMed Nucleotide Protein Genome Structure
Search PubMed for
Limits Preview/Index History Clipboard Details
Display AbstractPlus Show 20 Sort By
All: 1 Review: 0
1: Pathol Oncol Res. 2009 Feb 27. [Epub ahead of print]
Prognostic Significance of Phosphatase of Regenerating Liver-3 Expression in Ovarian Cancer.
Ren T, Jiang B, Xing X, Dong B, Peng L, Meng L, Xu H, Shou C.
Key laboratory of Carcinogenesis and Translational Research (Ministry of Education), Department of Biochemistry and Molecular Biology, Peking University School of Oncology, Beijing Cancer Hospital & Institute, 52 Fu, cheng Road, Beijing, 100, 142, C, hina.
Phosphatase of regenerating liver-3 (PRL-3) is overexpressed in several human cancers and associated with tumor progression, invasion and metastasis. However, the correlation between PRL-3 expression and clinical outcome in ovarian cancer has not been studied. In the present study, we investigated the expression of PRL-3 in 119

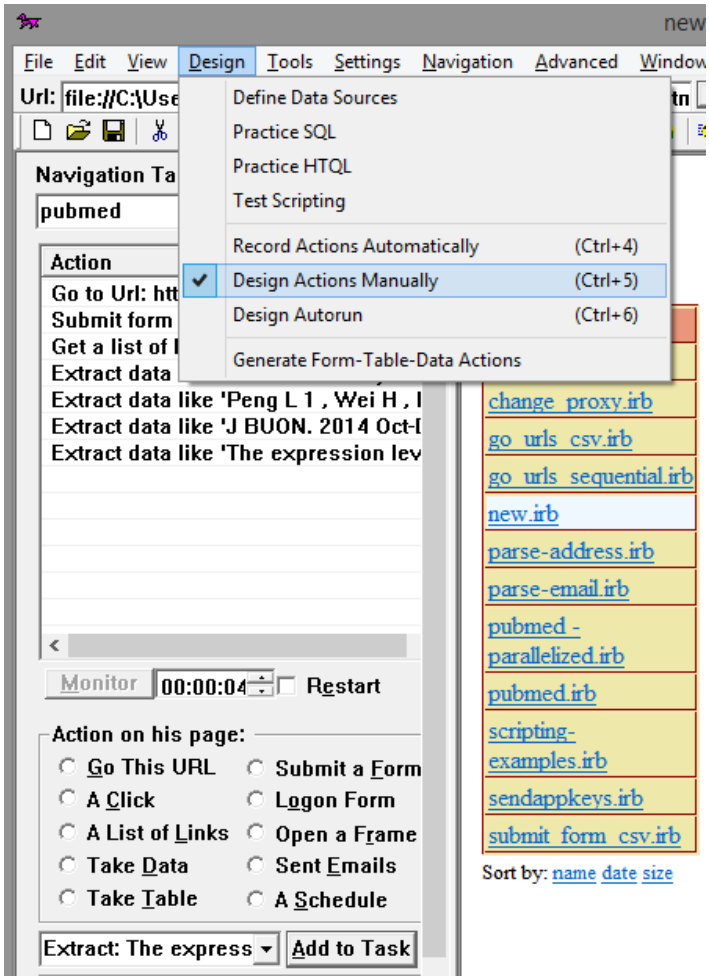
Press the [Run] button on the left panel to test-run this robot. You will see it does everything correctly. You may want to skip the following section "Create Actions Step-by-Step" if the recording can fit your need.

2.2. Design Actions Manually

An alternative way for robot creation is to create actions step by step. This allows advanced users to have a finer control over the creation of robot actions and to better cope with bugs appearing in the robot recording process. (The step-by-step creation is optimized for data recognition instead of recording speed by the recorder. This may solve some errors originated from ill-structured Web pages. Also note that the "Change Extraction …" menu item is another option if such error occurs.) We will describe instructions for each type of the action without much detailed explanations. Readers may follow the instructions and

re-create the example demonstrated in the previous robot recording section.

To create actions step-by-step, open the robot designer from menu: View → Design Actions Manually, as:



You can design ten types of robot actions using robot designer: Go to URL, A Click, A List of Links, Take Data, Take Table, Submit a Form, Logon Form, Open a Frame, Sent Emails and A Schedule, as listed in the designer panel. Detailed instruction follows:

A) Go to URL

i) Navigate to an URL in the browser window;
ii) Click on the 'Go This URL' radio button in the designer window;
iii) Click the [Add to Task] button.

B) A Click

i) Show a page in the browser window;
ii) Mark the link to be clicked in the browser window (right-click on the link but do

not select any option from the popup menu, or move the mouse above the link, press the left button, drag the link slightly, and release the left button);

iii) Click on the 'A Click' radio button in the designer window;

iv) Click the [Add to Task] button.

C) A List of Links

i) Show a page in the browser window;

ii) Mark one of the links to be followed in the browser window (right-click on the link but do not select any option from the popup menu, or move the mouse above the link, press the left button, drag the link slightly, and release the left button);

iii) Click on the 'A List of Links' radio button in the designer window;

iv) Click the [Add to Task] button.

D) Take Data

i) Show a page in the browser window;

ii) Mark the data to extract in the browser window;

iii) Click on the 'Take Data' radio button in the designer window;

iv) Click the [Add to Task] button.

E) Take Table

i) Show a page in the browser window;

ii) Mark a row of data to be extracted in the browser window;

iii) Click on the 'Take Table' radio button in the designer window;

iv) Click the [Add to Task] button.

F) Submit a Form

i) Show a page with forms in the browser window;

ii) Mark the button that can submit the form in the browser window (right-click on the link but do not select any option from the popup menu, or move the mouse above the link, press the left button, drag the link slightly, and release the left button);

iii) Click on the 'Submit a Form' radio button in the designer window;

iv) The designer window is switched to a form value window;

v) Enter a new name in the drop-down box under the 'Form Values:' (or you may select an existing name from the drop-down box);

vi) Click on the [Replace Form] button in the form value window.

vii) Press [OK] in the form value window;

viii) The form value window is switched to the designer window;
ix) Click the [Add to Task] button.

G) Logon Form

i) Show a page with logon forms in the browser window;
ii) Mark the button that can submit the form in the browser window (right-click on the link but do not select any option from the popup menu, or move the mouse above the link, press the left button, drag the link slightly, and release the left button);
iii) Click on the 'Logon Form' radio button in the designer window;
iv) The designer window is switched to a form value window;
v) Enter a new name in the drop-down box under the 'Form Values:' (or you may select an existing name from the drop-down box);
vi) Click on the [Replace Form] button in the form value window, as in the above figure.
vii) Press [OK] in the form value window;
viii) The form value window is switched to the designer window;
ix) Click the [Add to Task] button.

H) Open a Frame

i) Show an example page with multiple frames in the browser window;
ii) Mark any text in the target frame in the browser window;
iii) Click on the 'Open a Frame' radio button in the designer window;
iv) Click [Add to Task] button.

I) Sent Emails

v) To be discussed.

J) A Schedule

i) Click on the 'A Schedule' radio button in the designer window;
ii) Click the [Add to Task] button;
iii) Double-click the new 'Schedule tasks' action in the action list box;
iv) Press the [Insert] button on the page in the browser window;
v) Add new schedule tasks.

Examples are given below:

Scheduled Tasks

	Type	Interval	Base	Action	ActionType	Session
☐	DoEvery	1.5	Minute	check yahoo mails	CallPath	yahoomail1
☐	DoEvery	1.5	Minute	check gmails	CallPath	gmail1
☐	DoEvery	1.5	Minute	check hotmails	CallPath	hotmail1

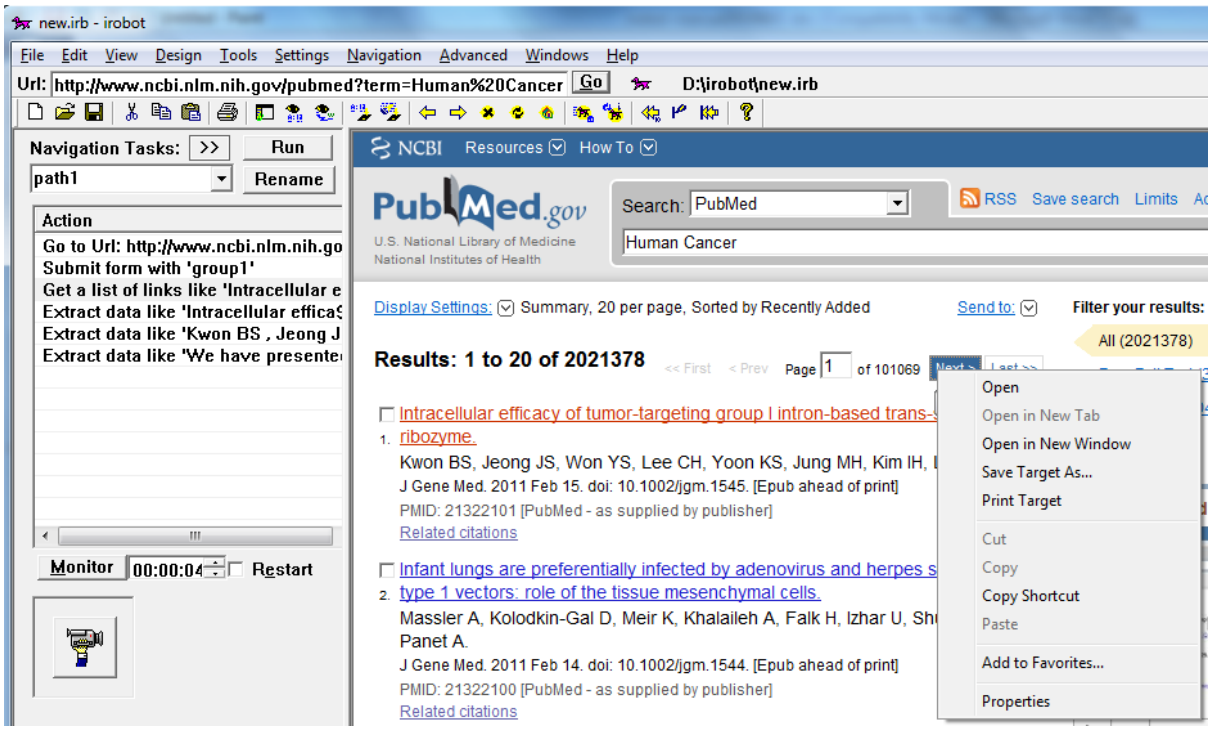
Modify Return Insert Delete

2.3. Repeat on Next Pages

The "new.irb" robot we have created can open every entry from a Pubmed result page automatically. Can it follow the Next pages continuously, and from each next page, repeat the actions it does on the first page? We will describe how to enable it in the following.

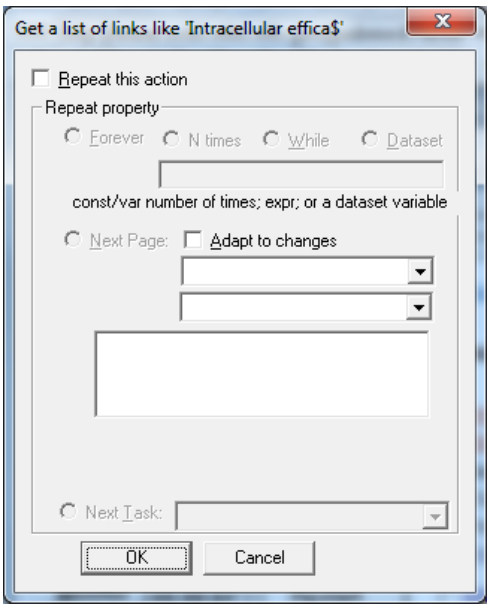
A) Design Repeat on Next Pages

1) Bring up a page including a list of pubmed-search results in the browser panel like:

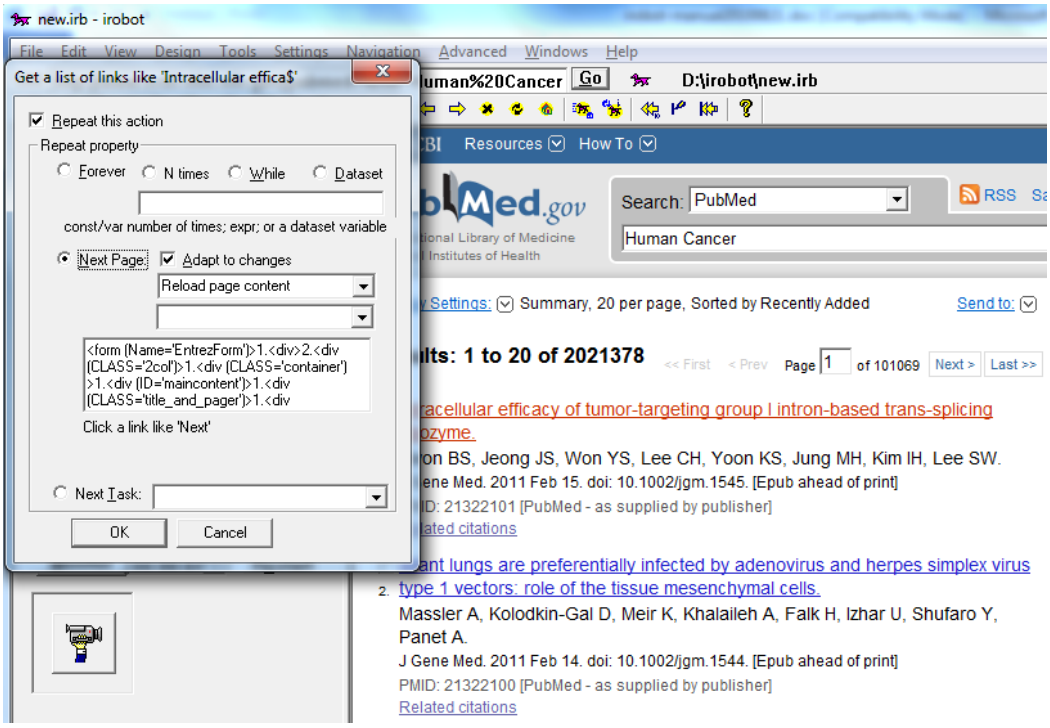

new.irb - irobot
File Edit View Design Tools Settings Navigation Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/pubmed?term=Human%20Cancer Go D:\irobot\new.irb
Navigation Tasks: >> Run
path1 Rename
Action
Go to Url: http://www.ncbi.nlm.nih.go
Submit form with 'group1'
Get a list of links like 'Intracellular e
Extract data like 'Intracellular effica
Extract data like 'Kwon BS , Jeong J
Extract data like 'We have presente
Monitor 00:00:04 Restart
NCBI Resources How To
PubMed.gov
U.S. National Library of Medicine
National Institutes of Health
Search: PubMed
Human Cancer
RSS Save search Limits
Display Settings: Summary, 20 per page, Sorted by Recently Added
Send to:
Filter your results:
All (2021378)
Results: 1 to 20 of 2021378
<< First < Prev Page 1 of 101069 Next >
1. Intracellular efficacy of tumor-targeting group I intron-based trans-
ribozyme.
Kwon BS, Jeong JS, Won YS, Lee CH, Yoon KS, Jung MH, Kim IH,
J Gene Med. 2011 Feb 15. doi: 10.1002/jgm.1545. [Epub ahead of print]
PMID: 21322101 [PubMed - as supplied by publisher]
Related citations
2. Infant lungs are preferentially infected by adenovirus and herpes s
type 1 vectors: role of the tissue mesenchymal cells.
Massler A, Kolodkin-Gal D, Meir K, Khalaileh A, Falk H, Izhar U, Sh
Panet A.
J Gene Med. 2011 Feb 14. doi: 10.1002/jgm.1544. [Epub ahead of print]
PMID: 21322100 [PubMed - as supplied by publisher]
Related citations
Open
Open in New Tab
Open in New Window
Save Target As...
Print Target
Cut
Copy
Copy Shortcut
Paste
Add to Favorites...
Properties

2) Mark the Next link from the Web page (right-click on the link but do not select any option from the popup menu, or move the mouse above the link, press the left button, drag the link slightly, and release the left button) (The highlighted Next link is shown in the above figure).

3) Right click on the action "Get a list of links like ***" from the left panel, and select the "Repeat Property …" from the popup manual. It shows a dialog:



4) Click on the [Repeat this action] check box, and click on the [Next Page] radio button from popup dialog, as in the following figure. There will be a query shown in the text box, and a message "Click a link like 'Next'" shown under the text box. Please make sure that the Next link in the Web page has been marked before you clicking on the [Next Page] radio button.


new.irb - irobot
Get a list of links like 'Intracellular effica$'
Repeat this action
Repeat property
Forever
N times
While
Dataset
const/var number of times; expr; or a dataset variable
Next Page:
Adapt to changes
Reload page content
Click a link like 'Next'
Next Task:
OK
Cancel
Search: PubMed
Human Cancer
Summary, 20 per page, Sorted by Recently Added
Send to:
Massler A, Kolodkin-Gal D, Meir K, Khalaileh A, Falk H, Izhar U, Shufaro Y, Panet A.
J Gene Med. 2011 Feb 14. doi: 10.1002/jgm.1544. [Epub ahead of print]
PMID: 21322100 [PubMed - as supplied by publisher]
Related citations

B) Try with Other Repeat Options

5) We are almost there. For most websites, we are already done and the robot can now click the next pages properly. However, for some JavaScript heavy websites, such as the Pubmed, the robot still refuses to go to the next page properly. We then need some trials and errors with the following options:

Option 1: Below the "Adapt to change," select "Reload page from URL" instead of the "Reload page content". This works if the URL can bring us back to the list-of-links page.

Option 2: select "Do NOT reload page & use URL". This works if the URL of the Next link can bring us to the next page directly.

Option 3: select "Do NOT reload page & click". Further, close the repeat property, and open the property page of the "Get a List of Links" action: Check the "To open in new browser". This usually works if the above two options failed.

Option 4 (rarely used): If option 3 failed and the pages are not opened in the new window, then try this option. Uncheck the "To open in new browser" in the property page, and keep the "Do NOT reload page & click" in the repeat property. Then right-click the "Get a List of Links" action and select "Events…". Add a new event, Event: *Completed each tuple; Value: GoBack(). This option will tend to fail if any link is not opened correctly. So try NOT to use this option. It is more likely that you need to change the query of the repeat property to make it work; read further.

For Pubmed, the option currently working is Option 3.

Tip 1: To save time on testing the repeat property, you can temporarily add an event to the "Get a List of Links" action, with Event: After each tuple, Condition: Tuple>1, Return: End this page. This will allow the action to open only two links in each page before clicking the next.

Tip 2: Sometimes, the robot goes only to the 2nd page, and comes back to the 1st page after the 2nd page is completed. This is because the HTQL query in the Repeat property is not robust enough. You then need to manually find a better query that can consistently find the Next link. A typical query that works is:

<a (tx like 'Next%')>

So try this query in the Repeat property if there is a Next link on the page.

6) Click the [OK] button to close the Repeat property dialog. Now we can run the robot, and it will open each link page and follow Next pages automatically.

2.4. Open Links in New Windows

The "new.irb" robot we have created can open every entry from a Pubmed result page automatically. Sometimes, the robot will try to reload the table page before going to the next pages. You can let it run more efficiently with a little tweak—to open links in a new window, and click the next link without reloading the table page. You can do it with the following changes (also refer option 3 in Section 2.3 and the "pubmed.irb" online).

1) In the property of the "Get a list of links" action, check the checkbox after the "To open in new browser";

2) Change the "Before each click:" to "do NOT reload & click it";

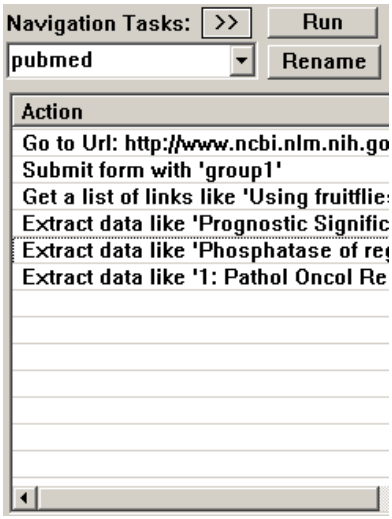
3) In its repeat property, select "do NOT reload page & click" from the dropdown list.

Now you will see the robot will open each link in a new window, and go to the next pages without reloading the content thus become more efficient.

2.5. Create Variables

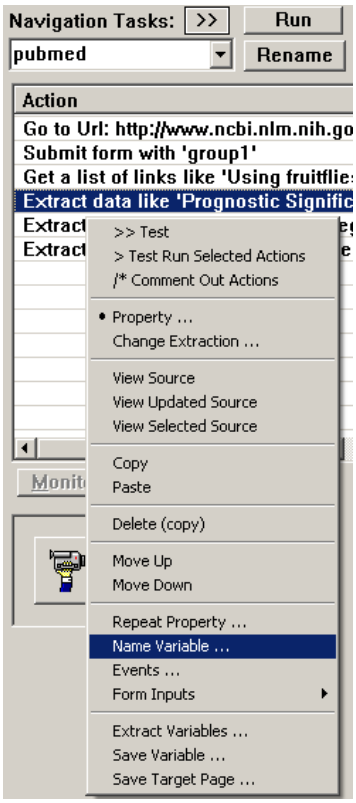
Now we have created the "new.irb" robot to do literature search and Web navigations. Our goal, however, is to extract the literature information and save them into a database. To achieve this goal, we still need to: 1) extract target Web information during robot navigation, 2) define variables to store the extracted data, and 3) save the variable data into a local database. We will illustrate how to define variables to store the extracted abstract author, title and text in this section.

From the "pubmed" task created previously (also shown below) we will define four variables: *AbstractURL*, *Title*, *Abstract*, and *Journal* for the 3rd, 4th, 5th and 6th action, respectively.

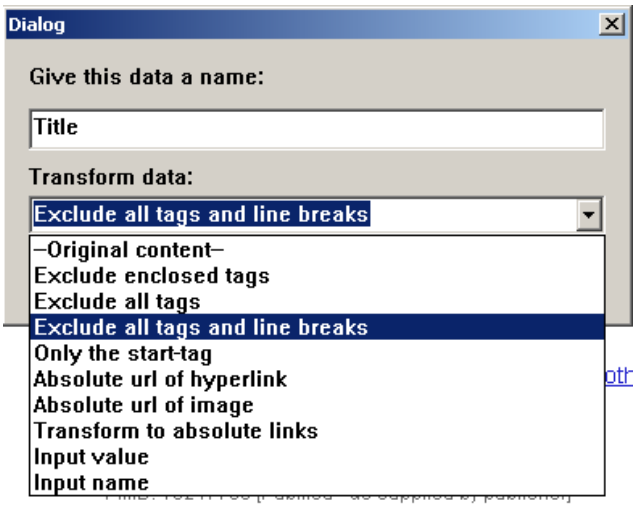


A) Create Variables for Extraction Data

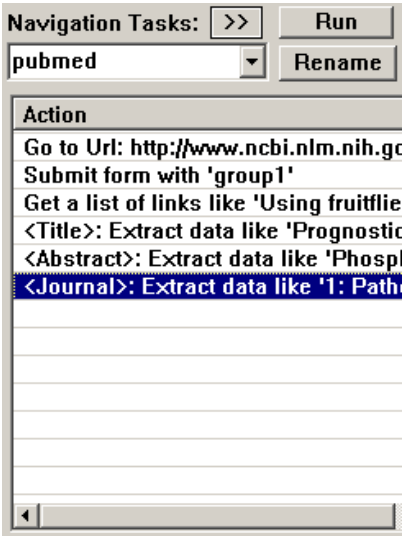
Right-click the forth action: "Extract data like 'Analysis of ***'", and select the "Name Variable …" menu item, as below.



It shows a dialog box as in the following. Give it a name "Title", select a data transformation option: "Exclude all tags and line breaks", and press [OK].

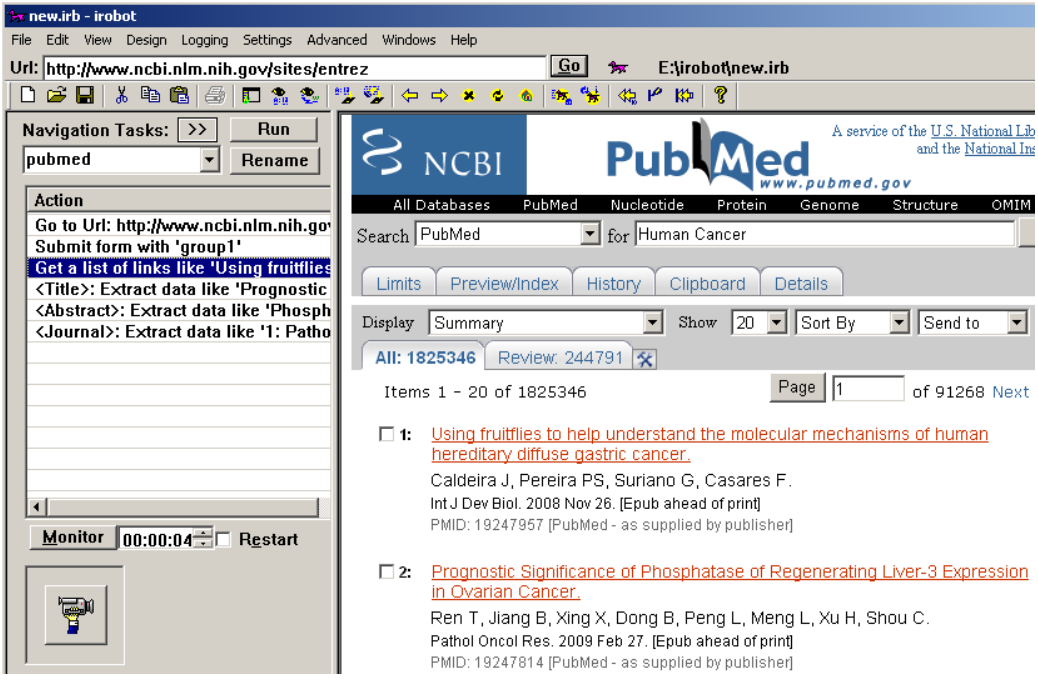


Similarly, define Abstract and Journal for the 5 and 6th action. The "pubmed" task becomes:

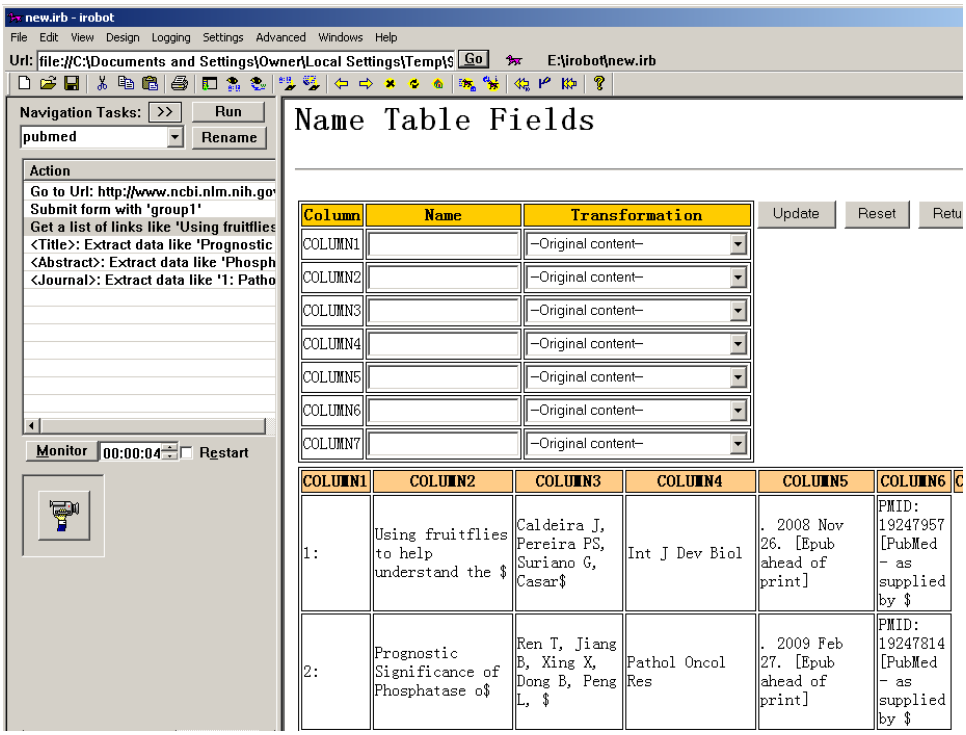


B) Create Variables for Table Data

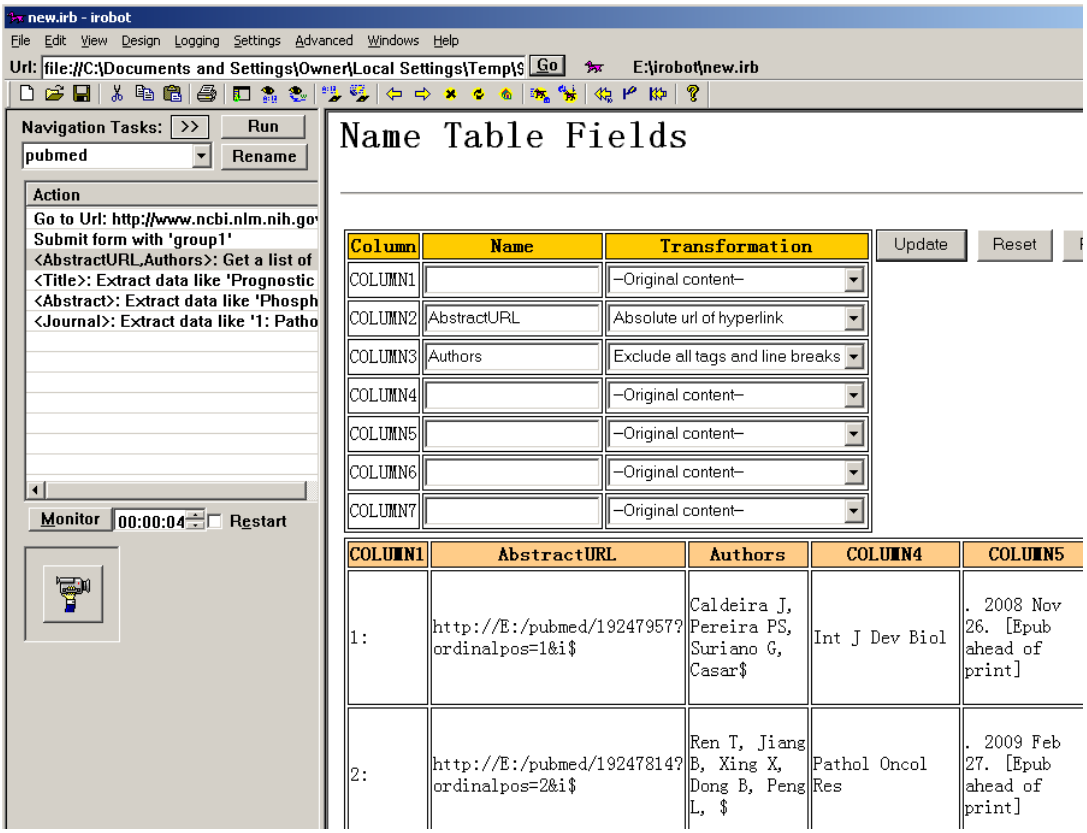
Finally, we are going to define the AbstractURL from the 3rd action, which is more complex. Bring up a result page including a list of entries from the pubmed search in the browser panel like:


new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: http://www.ncbi.nlm.nih.gov/sites/entrez
Go
E:\irobot\new.irb
Navigation Tasks: >>
Run
pubmed
Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
Get a list of links like 'Using fruitflies
<Title>: Extract data like 'Prognostic
<Abstract>: Extract data like 'Phosph
<Journal>: Extract data like '1: Patho
Monitor 00:00:04 Restart
NCBI
PubMed
www.pubmed.gov
All Databases PubMed Nucleotide Protein Genome Structure OMIM
Search PubMed for Human Cancer
Limits Preview/Index History Clipboard Details
Display Summary Show 20 Sort By Send to
All: 1825346 Review: 244791
Items 1 - 20 of 1825346
Page 1 of 91268 Next
1: Using fruitflies to help understand the molecular mechanisms of human hereditary diffuse gastric cancer.
Caldeira J, Pereira PS, Suriano G, Casares F.
Int J Dev Biol. 2008 Nov 26. [Epub ahead of print]
PMID: 19247957 [PubMed - as supplied by publisher]
2: Prognostic Significance of Phosphatase of Regenerating Liver-3 Expression in Ovarian Cancer.
Ren T, Jiang B, Xing X, Dong B, Peng L, Meng L, Xu H, Shou C.
Pathol Oncol Res. 2009 Feb 27. [Epub ahead of print]
PMID: 19247814 [PubMed - as supplied by publisher]

Then right-click the 3rd action "Get a list of links like ***", and select the "Name Variables …" menu item. It shows:


new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: file://C:\Documents and Settings\Owner\Local Settings\Temp\$ Go E:\irobot\new.irb
Navigation Tasks: >> Run
pubmed Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
Get a list of links like 'Using fruitflies
<Title>: Extract data like 'Prognostic
<Abstract>: Extract data like 'Phosph
<Journal>: Extract data like '1: Patho
Monitor 00:00:04 Restart
Name Table Fields
Column Name Transformation Update Reset Retu
COLUMN1 —Original content—
COLUMN2 —Original content—
COLUMN3 —Original content—
COLUMN4 —Original content—
COLUMN5 —Original content—
COLUMN6 —Original content—
COLUMN7 —Original content—
COLUMN1 COLUMN2 COLUMN3 COLUMN4 COLUMN5 COLUMN6
1: Using fruitflies to help understand the $ Caldeira J, Pereira PS, Suriano G, Casar$ Int J Dev Biol . 2008 Nov 26. [Epub ahead of print] PMID: 19247957 [PubMed - as supplied by $
2: Prognostic Significance of Phosphatase o$ Ren T, Jiang B, Xing X, Dong B, Peng L, $ Pathol Oncol Res . 2009 Feb 27. [Epub ahead of print] PMID: 19247814 [PubMed - as supplied by $

As you can see, the search result page serves as a sample page for data columns. Put a name "AbstractURL" after the "COLUMN2", select a transformation "Absolute url of hyperlink" after it, and click the [Update]. It shows:

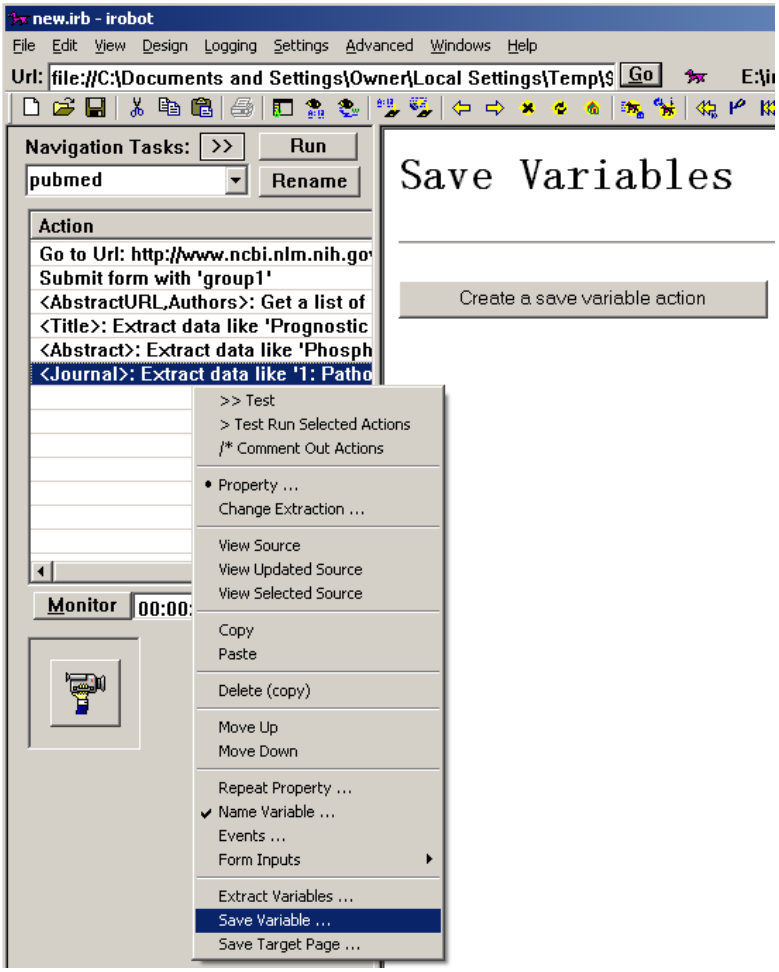

new.irb - irobot
File Edit View Design Logging Settings Advanced Windows Help
Url: file://C:\Documents and Settings\Owner\Local Settings\Temp\$ Go E:\irobot\new.irb
Navigation Tasks: >> Run
pubmed Rename
Action
Go to Url: http://www.ncbi.nlm.nih.gov
Submit form with 'group1'
<AbstractURL,Authors>: Get a list of
<Title>: Extract data like 'Prognostic
<Abstract>: Extract data like 'Phosph
<Journal>: Extract data like '1: Patho
Monitor 00:00:04 Restart
Name Table Fields
Column Name Transformation Update Reset
COLUMN1 —Original content—
COLUMN2 AbstractURL Absolute url of hyperlink
COLUMN3 Authors Exclude all tags and line breaks
COLUMN4 —Original content—
COLUMN5 —Original content—
COLUMN6 —Original content—
COLUMN7 —Original content—
COLUMN1 AbstractURL Authors COLUMN4 COLUMN5
1: http://E:/pubmed/19247957?ordinalpos=1&i$ Caldeira J, Pereira PS, Suriano G, Casar$ Int J Dev Biol . 2008 Nov 26. [Epub ahead of print]
2: http://E:/pubmed/19247814?ordinalpos=2&i$ Ren T, Jiang B, Xing X, Dong B, Peng L, $ Pathol Oncol Res . 2009 Feb 27. [Epub ahead of print]

Notice that URLs are shown in the sample table in the lower half of the page. Similarly, we may define the *Authors* for "COLUMN3", with the option "Exclude all tags and line breaks".

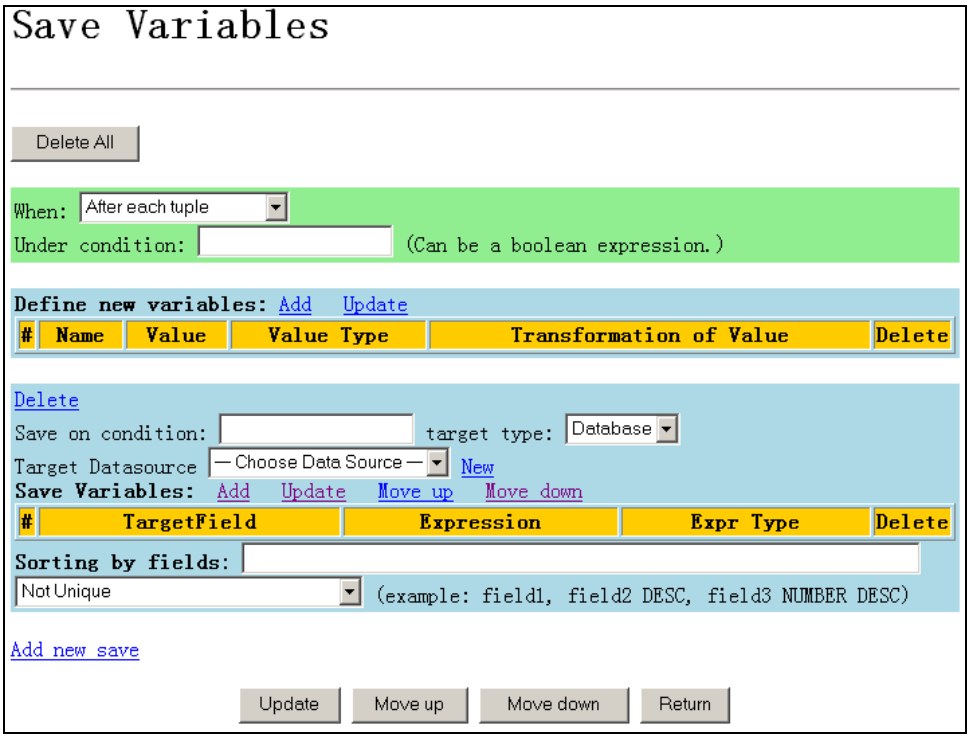
Now click [Return]. We have finished defining the five variables.

2.6. Save Variables into Databases

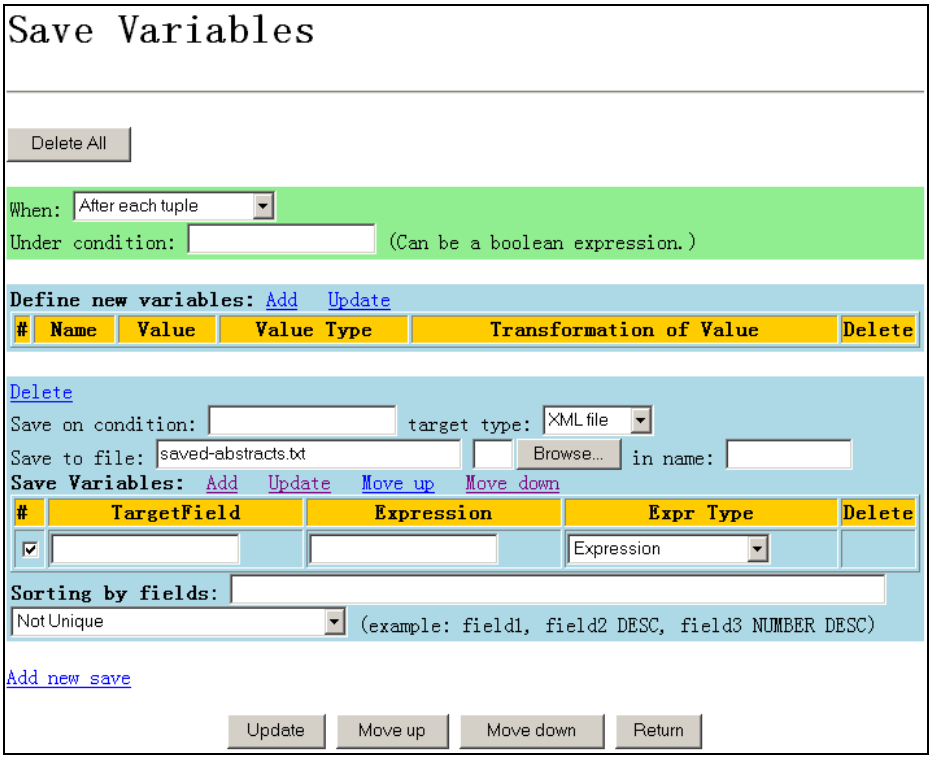
We are going to save the literature information into an XML file database including the AbstractURL, Title, Journal, and Abstract variables. Right-click on the 6th action "Abstract: ***", and select the "Save Variable …" menu item, it shows:



Click on the [Create a save variable action], it shows:

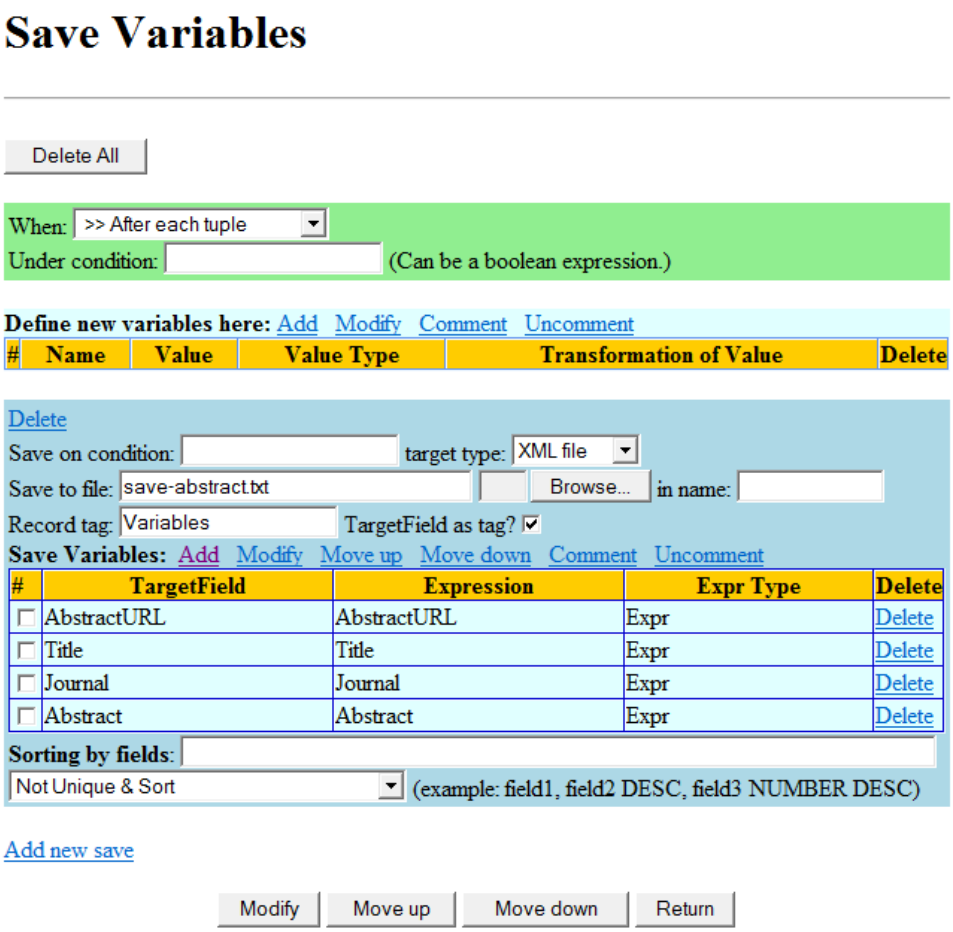

Save Variables
Delete All
When: After each tuple
Under condition: (Can be a boolean expression.)
Define new variables: Add Update
Name Value Value Type Transformation of Value Delete
Delete
Save on condition: target type: Database
Target Datasource — Choose Data Source — New
Save Variables: Add Update Move up Move down
TargetField Expression Expr Type Delete
Sorting by fields:
Not Unique (example: field1, field2 DESC, field3 NUMBER DESC)
Add new save
Update Move up Move down Return

Select the "XML file" from the "target type: [Database]" drop-down list, enter "saved-abstracts.txt" in the "Save to file: [ ]" input box (it will appear once the "XML file" is selected), and click the "Add" following the "Save Variables:". It shows:

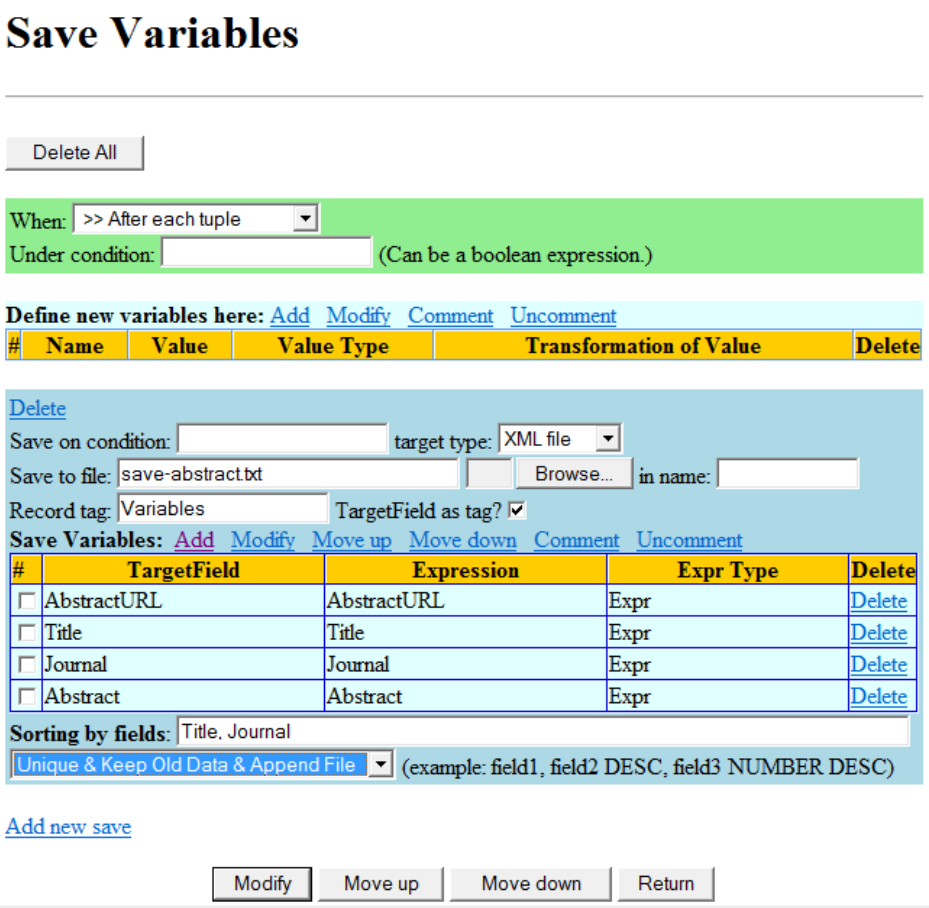

Save Variables
Delete All
When: After each tuple
Under condition: (Can be a boolean expression.)
Define new variables: Add Update
Name Value Value Type Transformation of Value Delete
Delete
Save on condition: target type: XML file
Save to file: saved-abstracts.txt Browse... in name:
Save Variables: Add Update Move up Move down
TargetField Expression Expr Type Delete
Expression
Sorting by fields:
Not Unique (example: field1, field2 DESC, field3 NUMBER DESC)
Add new save
Update Move up Move down Return

Enter "AbstractURL" in both the "TargetField" and the "Expression" boxes, click "Add";
Enter "Title" in both the "TargetField" and the "Expression" boxes, click "Add";
Enter "Journal" in both the "TargetField" and the "Expression" boxes, click "Add";
Enter "Abstract" in both the "TargetField" and the "Expression" boxes, click [Update] button;

It shows:


Save Variables
Delete All
When: >> After each tuple
Under condition: (Can be a boolean expression.)
Define new variables here: Add Modify Comment Uncomment
Name Value Value Type Transformation of Value Delete
Delete
Save on condition: target type: XML file
Save to file: save-abstract.txt Browse... in name:
Record tag: Variables TargetField as tag?
Save Variables: Add Modify Move up Move down Comment Uncomment
TargetField Expression Expr Type Delete
AbstractURL AbstractURL Expr Delete
Title Title Expr Delete
Journal Journal Expr Delete
Abstract Abstract Expr Delete
Sorting by fields:
Not Unique & Sort (example: field1, field2 DESC, field3 NUMBER DESC)
Add new save
Modify Move up Move down Return

We would like to sort the output by Title and Journal without duplication. So we set the "Title, Journal" after the "Sorting by fields:" and select the [Unique & Keep Old Data & Append File] from the drop down list. It is like this:


Save Variables
Delete All
When: >> After each tuple
Under condition: (Can be a boolean expression.)
Define new variables here: Add Modify Comment Uncomment
Name Value Value Type Transformation of Value Delete
Delete
Save on condition: target type: XML file
Save to file: save-abstract.txt Browse... in name:
Record tag: Variables TargetField as tag?
Save Variables: Add Modify Move up Move down Comment Uncomment
TargetField Expression Expr Type Delete
AbstractURL AbstractURL Expr Delete
Title Title Expr Delete
Journal Journal Expr Delete
Abstract Abstract Expr Delete
Sorting by fields: Title, Journal
Unique & Keep Old Data & Append File (example: field1, field2 DESC, field3 NUMBER DESC)
Add new save
Modify Move up Move down Return

Click [Return] and we are done.

Now run the robot, and open the "saved-abstracts.txt" file in the IROBOT installation directory. You can see the abstract information is saved correctly and there is no duplication.

2.7. Draw Data from Databases

We are going to automate the literature search with multiple keywords. For example, we have three candidate keywords and we want to submit them for Pubmed search in batch:

Human cancer

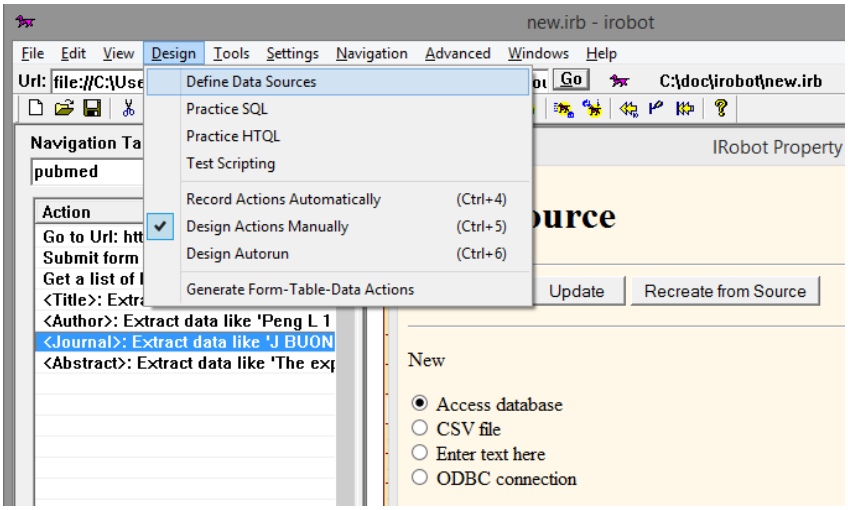
Tumor

Breast cancer

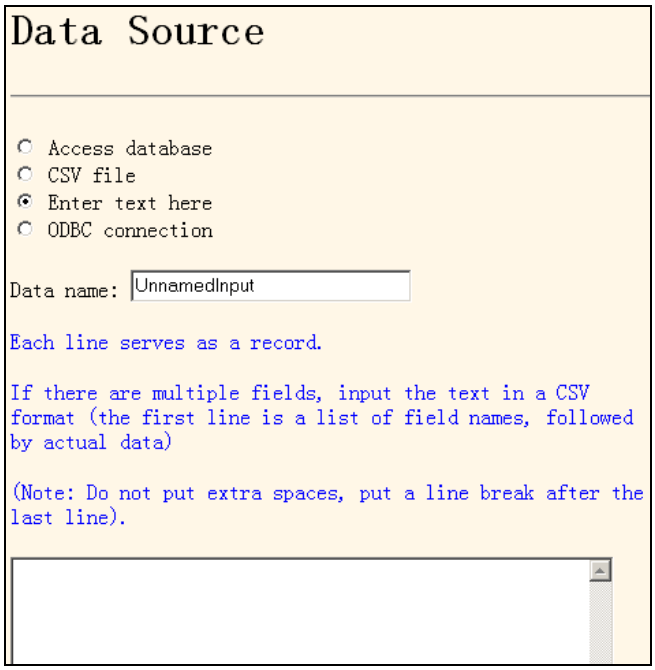
We can do this in two methods: 1) setup a connection between the data source and the form inputs, and let iRobot automatically draw input from the data source; and 2) load the data into a dataset variable, manually repeat on the dataset, and use dataset fields as variables for form inputs. The second method requires some programming skill and will be introduced later in Section 5.3 Dataset Variables and Functions. This section introduces the first method with two steps: A) Define New Data Sources, and B) Connect Robot Input with Data Sources. This method is relatively simple and does not require any programming skill.

A) Define New Data Sources

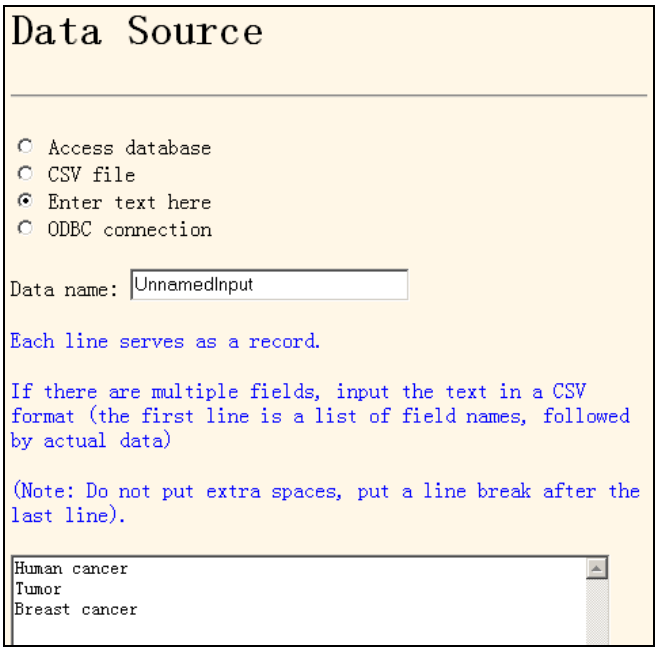
First, we will create a *data source* to include the above three entries. Select the main menu Design → Define Data Sources, the following page is shown.



Select the "Enter text here" radio button, which means that the data will be drawn from some text inputs. If you have keywords in an Access database, or in a CSV file, choose the corresponding radio button instead. After clicking the "Enter text here" radio button, it shows:


Data Source
Access database
CSV file
Enter text here
ODBC connection
Data name: UnnamedInput
Each line serves as a record.
If there are multiple fields, input the text in a CSV format (the first line is a list of field names, followed by actual data)
(Note: Do not put extra spaces, put a line break after the last line).

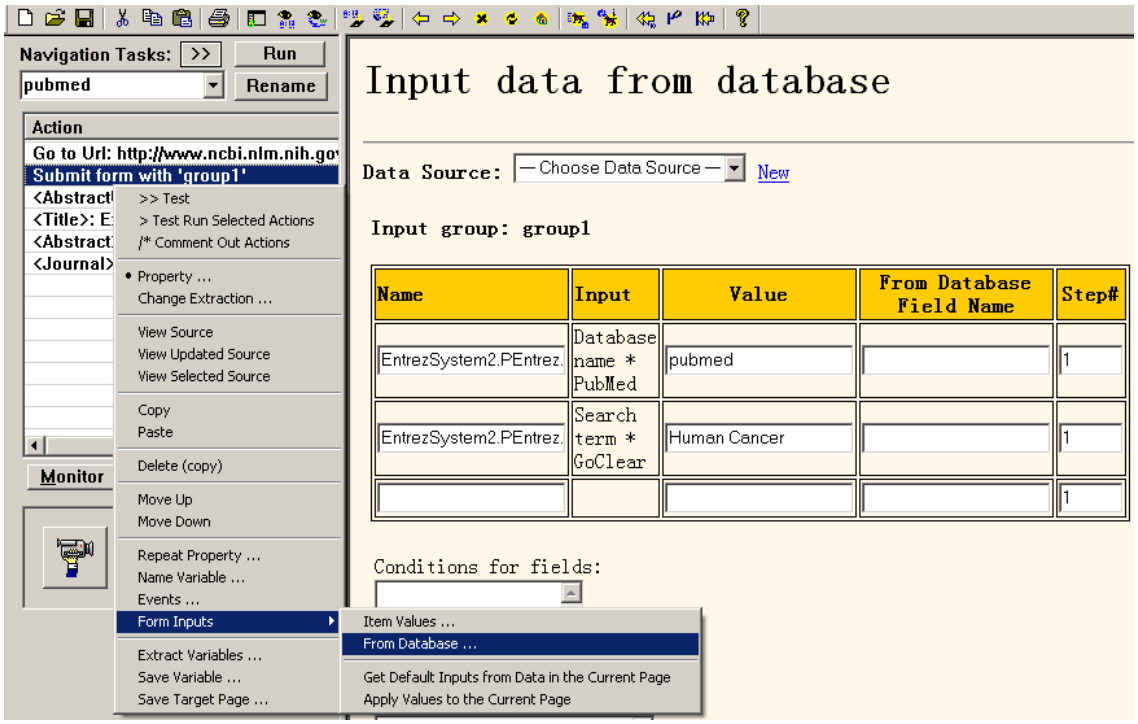
Give the data source a name, default as "UnnamedInput", and fill the keywords in the text box, each keyword in a line:

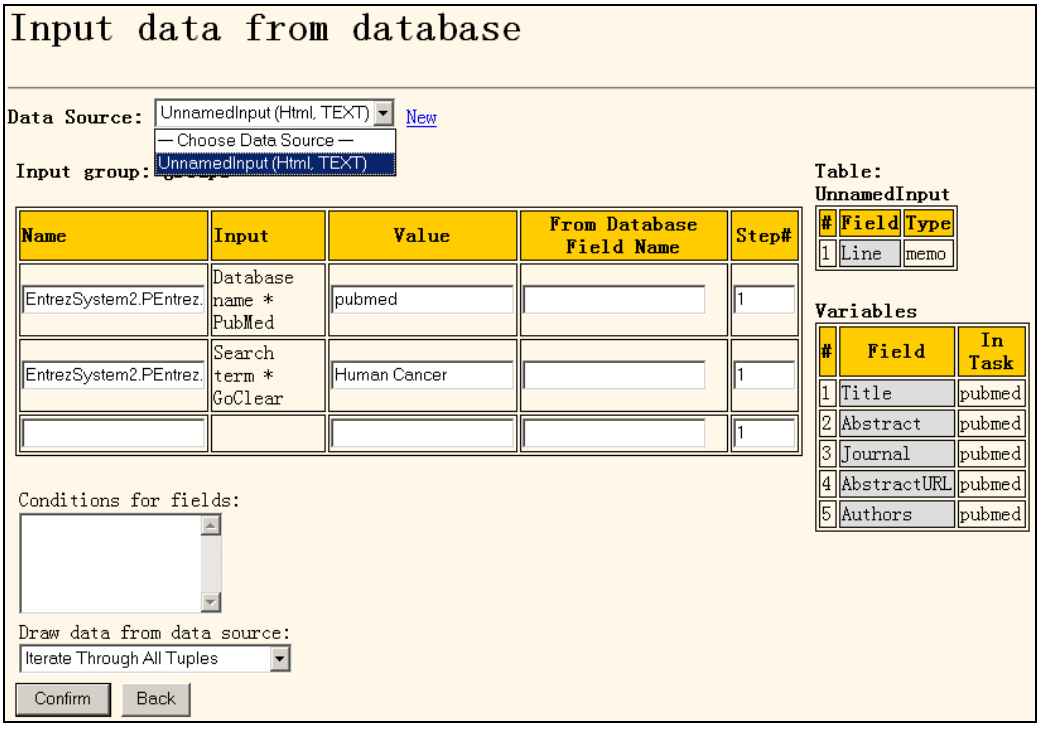

Data Source
Access database
CSV file
Enter text here
ODBC connection
Data name: UnnamedInput
Each line serves as a record.
If there are multiple fields, input the text in a CSV format (the first line is a list of field names, followed by actual data)
(Note: Do not put extra spaces, put a line break after the last line).
Human cancer
Tumor
Breast cancer

Then press [Next], a confirmation page is shown, click on the 'Done' link, and we return to the original data source page.

B) Connect Robot Input with Data Sources

Next, we will show how to modify the previously created Pubmed robot to submit searches using values in the "UnnamedInput" data source we have just created. Right-click on the second action "Submit form with group1" and from the popup menu, select "Form Inputs → From Database …", as in the following:

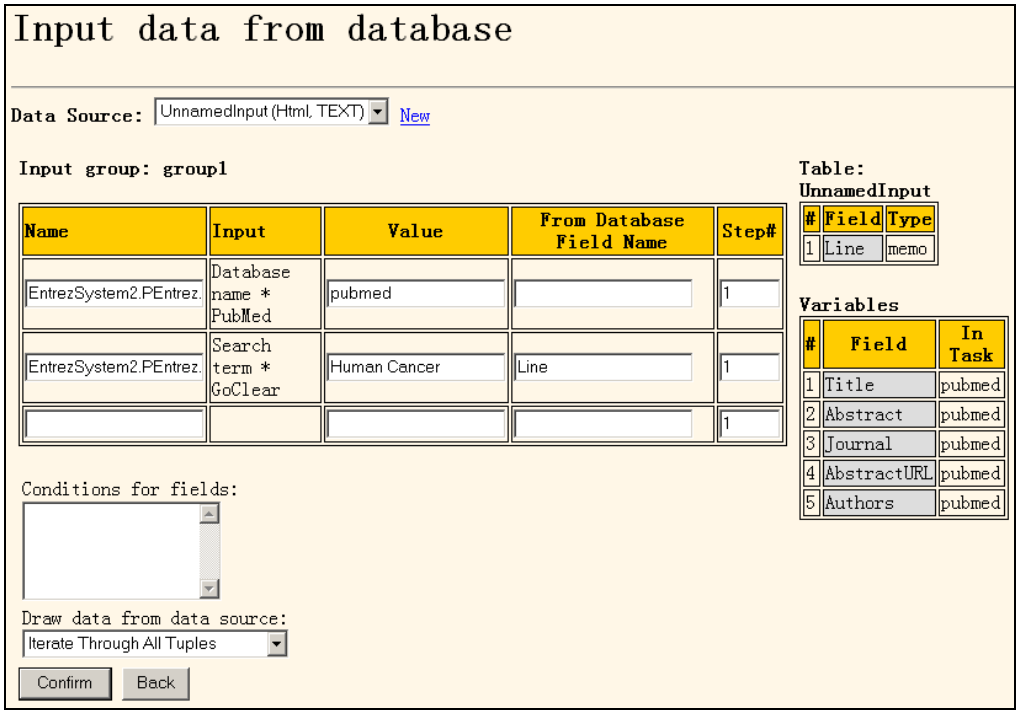




Now select the "UnnamedInput" from the dropdown menu following the "Data Source", as above, and the "Table: UnnamedInput" is shown on the right. We see the only field in the "UnnamedInput" table is the "1 | Line | memo".

We are going to submit the "Line" field from the database to the "EntrezSystem2.* | Search term * GoClear | Human cancer" input of the form. So simply put the field name "Line" in

the corresponding input box as shown below. (Or you can first click the input box, then click the "Line" field under the Table: UnnamedInput, and the "Line" field will automatically set in the input box):



Finally, click the [Confirm] button. We are done with the batch definition.

Now re-run the Pubmed robot, and we will see it continuously draw input from our list of keywords and do the search with each keyword.

An alternative method to connect robot input with data sources is to use dataset variables. Refer section 5.3 Dataset Variables and Functions for this.

2.8. Scheduled Run Tasks

We can easily configure the Pubmed robot to run automatically every few hours. First, type in a new task name after the "Task" and click the "New Task" button. Then, open the design view from menu: View → Design Actions Manually. From the design panel, add an action "A Schedule" and double click the action. From the web page, you can add a schedule to call the "pubmed" task every few hours.

3. Bug Fixing

It is well understood that HTML pages change frequently and any rules-based system may fail when substantial changes have occurred. Fortunately, IRobot allows you to easily divide and conquer complex navigation tasks into smaller pieces. Here are some tips that

may help you fix any bugs:

1) Keep in mind that each action or sequence of actions in IRobot can be tested independently. So first test each action thoroughly and make sure it is performing as expected. Subsection 3.1 "Debugging Robot Errors" shows some useful tools for testing robot actions and pinpointing error actions.
2) A typical cause of error is that robot actions are not robust enough, and they no longer work with new Web pages. Subsection 3.2 "Repairing Robot Actions" demonstrates some simple tools to fixing them.
3) However, if you have pinpointed the error, and the above tools cannot solve the problem, you may need to manually recompose the Query in the Property of each action, and make them robust across all pages. A typical method is to change their extraction query from multiple pages, compare the different queries it generated automatically, and come up with your best query.

The remaining section guides you through steps for checking robot errors, and tools for fixing the robots.

You may need to have a basic understanding of how IRobot works before fixing bugs. The basic logic of a robot task is a recursive invocation of a set of actions. In other words, actions are not working sequentially, but *recursively*. For example, if there are three actions A1, A2, A3 in a robot task T, then, you may think of task T as a function like the following:

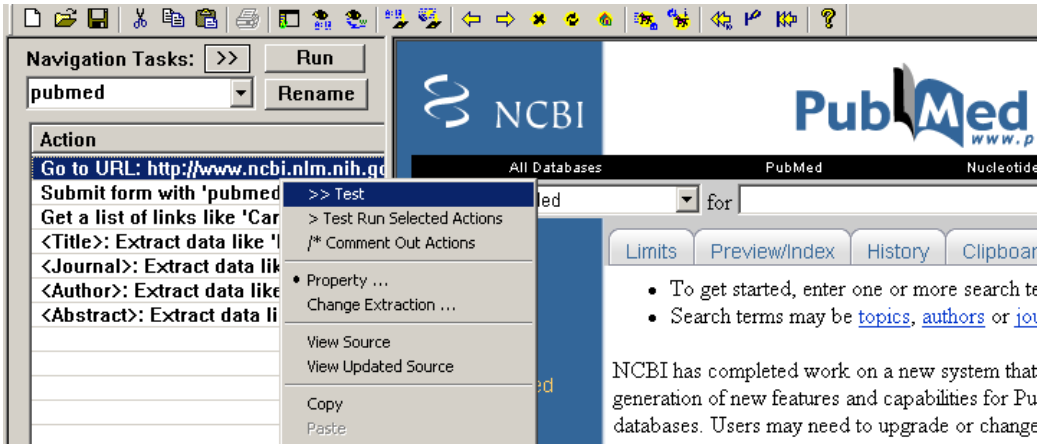
```
Function T()
    For each tuple a1 in A1
        Do some computation on a1
        For each tuple a2 in A2
            Do some computation on a2
            For each tuple a3 in A3
                Do some computation on a3
        End
    End
End
```

Without understanding this recursive logic, you may find some weird behavior from your robots. Note that if no tuple is found from the current action, the rest of the actions will NOT be executed. An exception is for the "Extract Data" action as the current action, where no matter the data is found from the page or not, the next action will be invoked. (Tip: If "A Click" action cannot find the target link but you still want it to continue anyway, you can use an event and set its return value as "Success".)

3.1. Debugging Robot Errors

A) Test a Single Action

The simplest way to test a robot action is to just right click on the robot action and select the "Test" menu item.

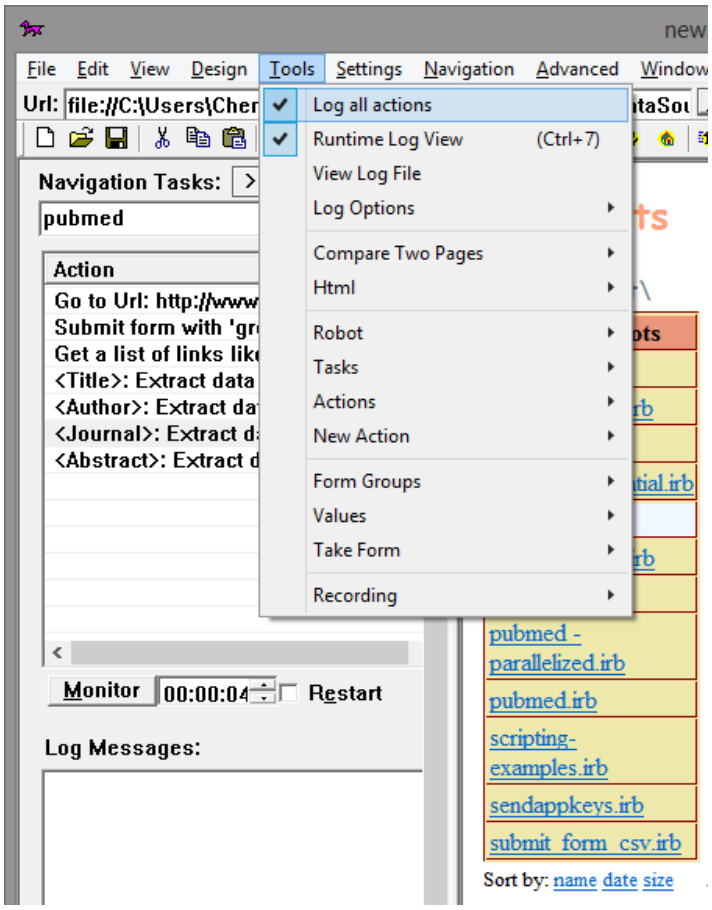


B) Test Run Selected Actions

The "Test" option simply checks if the action is working as expected. It does not evaluate any variables, events, or task calls associated with the action. In order to check these associated actions, you can use the other menu item: "Test Run Selected Actions". This can be combined with the following action log method to check if variables are assigned correctly, events are executed in the right order, or task calls have been invoked as wanted.

C) View the Log File

The best way to debug a robot is to open its action log. You can log robot actions by opening the main menu → Tools → Log all actions. Then run the robot, and open the "IROBOT\system\action.log" file.



It may give you something like this:

```
0|2007/08/24 19:47:43| ═══════ Start: ═══════
16|2007/08/24 19:47:43| [pubmed]*1(0)* : Url : http://www.ncbi.nlm.nih.gov/entrez/query.fcgi?db=PubMed&itool=toolbar
16|2007/08/24 19:47:46| [pubmed]*2(0)* : Form : pubmed-key
16|2007/08/24 19:47:51| [pubmed]*3(0)* : Table : <FORM (Name='EntrezForm')>1.<TABLE (ID='resultview')>1.<TABLE
16|2007/08/24 19:47:54| [pubmed]*4(0)*Title : Item : <FORM (Name='EntrezForm')>1.<TABLE (ID='resultview')>1.<TAB
2|2007/08/24 19:47:54|          Title:="<H2>The utility of focused assessment with sonography for trauma as a triage tool i
2|2007/08/24 19:47:54|          Title:="The utility of focused assessment with sonography for trauma as a triage tool in mul
16|2007/08/24 19:47:54| [pubmed]*5(0)*Journal : Item : <FORM (Name='EntrezForm')>1.<TABLE (ID='resultview')>1.<T
2|2007/08/24 19:47:54|          Journal:="<SPAN class=ti><INPUT id=UidCheckBox type=checkbox value=17715308 na
2|2007/08/24 19:47:54|          Journal:="1: J Ultrasound Med. 2007 Sep;26(9):1149-56."
```

You can then check if actions are taking the right data, and variables are correctly assigned.

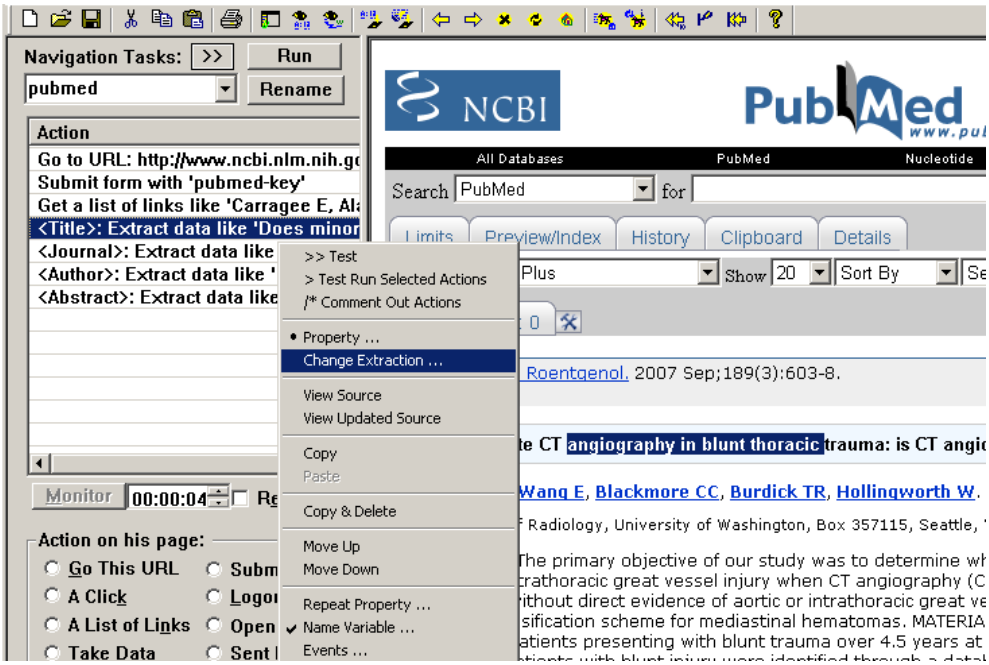
3.2. Repairing Robot Actions

If errors are detected, you may want to repair the robot actions or recreate the action. Repairing is desirable if you want to preserve the variables or events that are associated with an action instead of recreating everything from the scratch.

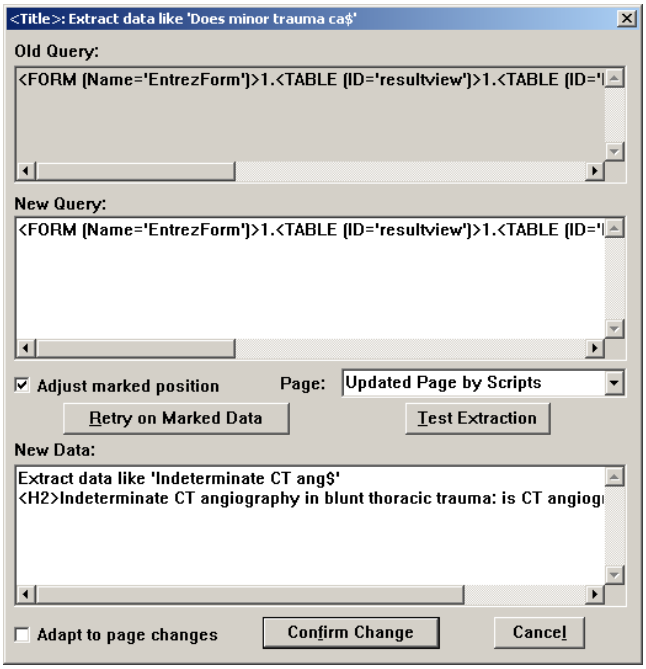
A) Change the Extraction Query by Wizard

The most common error is because the Web page has changed substantially and the action can no longer take the right data from the page. In this case, the first choice is to use the "Change Extraction …" wizard from the action menu. But before doing so, you first *mark*

the correct data in the Web page, and the "Change Extraction …" wizard will try to relocate data based on your marked position and regenerate a data extraction query in HTQL.

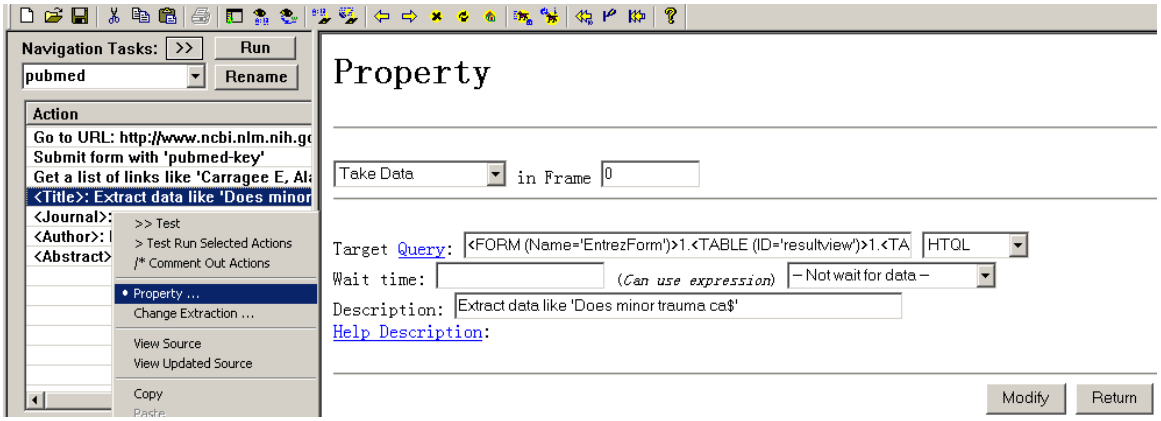


There are multiple options in the "Change Extraction …" wizard as shown in the following figure. Both old and new queries will be shown in the wizard interface, as well as the data being extracted. If the data is extracted not as expected, you can 1) change slightly the marked positions in the page and retry on the marked data; 2) change some options in the wizard panel and regenerate the query; and 3) if you are familiar with HTQL syntax, you may change the new query expression manually and test it.



B) Check the Action Property

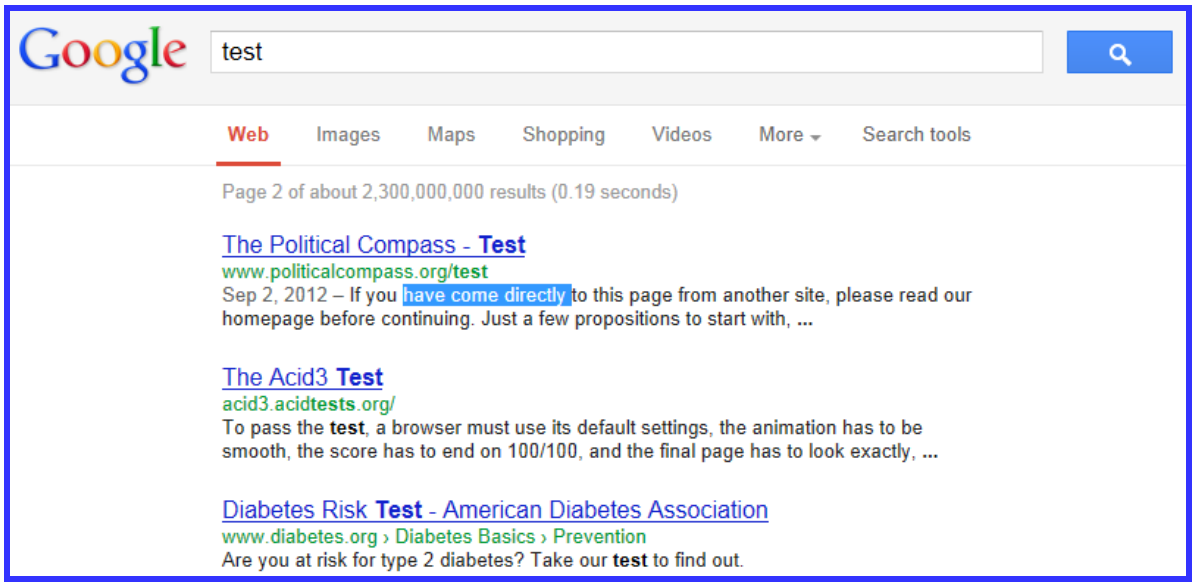
The wizard simplest changes the data query in the action property. You may change the query directly without invoking the wizard. The action property also includes other parameters that are not present in the wizard. Right click on the action and select the "Property …", you will able to see the target query in the property page. In this property page, you may also customize other options for the action, for example, you may use queries in a variable instead of a constant HTQL expression.



You may click the "Query" link on the property page to practice the use of different HTQL queries. Remember to navigate to the target Web page before opening the property page, and all queries will be targeted to the target page.

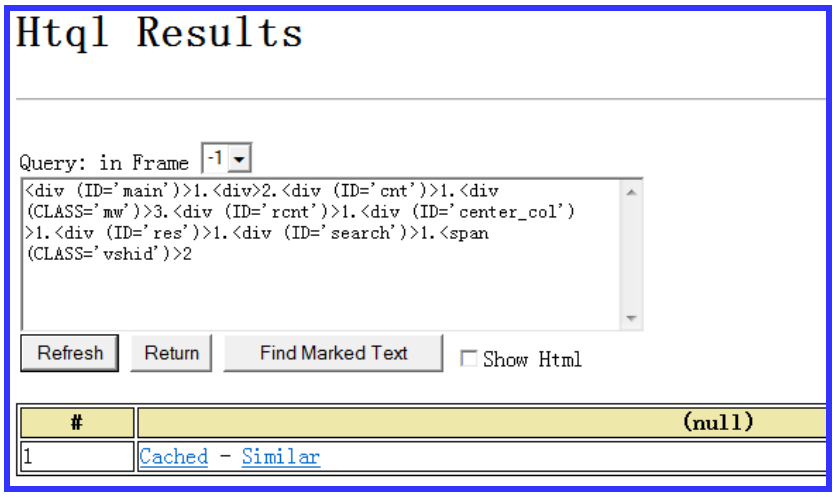
3.3. Change HTQL Manually

Automatically generated HTQL query is usually not very robust. It is recommended to check every query manually and make sure it can work across different pages. Detailed HTQL syntax can be found at http://htql.net. Here we will walk through an example to extract Google search results, which is usually difficult to do automatically. The following screenshot shows the sample page.


Google
test
Web
Images
Maps
Shopping
Videos
More
Search tools
Page 2 of about 2,300,000,000 results (0.19 seconds)
The Political Compass - Test
www.politicalcompass.org/test
Sep 2, 2012 – If you have come directly to this page from another site, please read our homepage before continuing. Just a few propositions to start with, ...
The Acid3 Test
acid3.acidtests.org/
To pass the test, a browser must use its default settings, the animation has to be smooth, the score has to end on 100/100, and the final page has to look exactly, ...
Diabetes Risk Test - American Diabetes Association
www.diabetes.org › Diabetes Basics › Prevention
Are you at risk for type 2 diabetes? Take our test to find out.

A) Practice HTQL

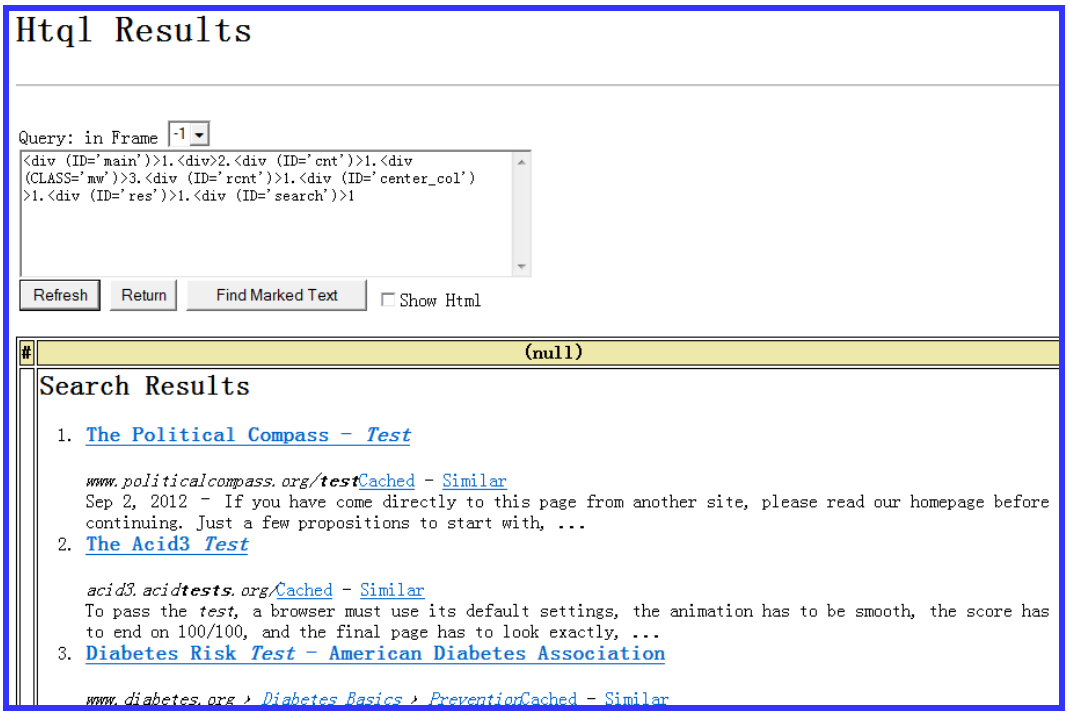
We mark the "**have come directly**" on the page, and select "Manual -> Design -> Practice HTQL", but it does not work as expected: it extracted the "Cached – Similar" instead of the description text.



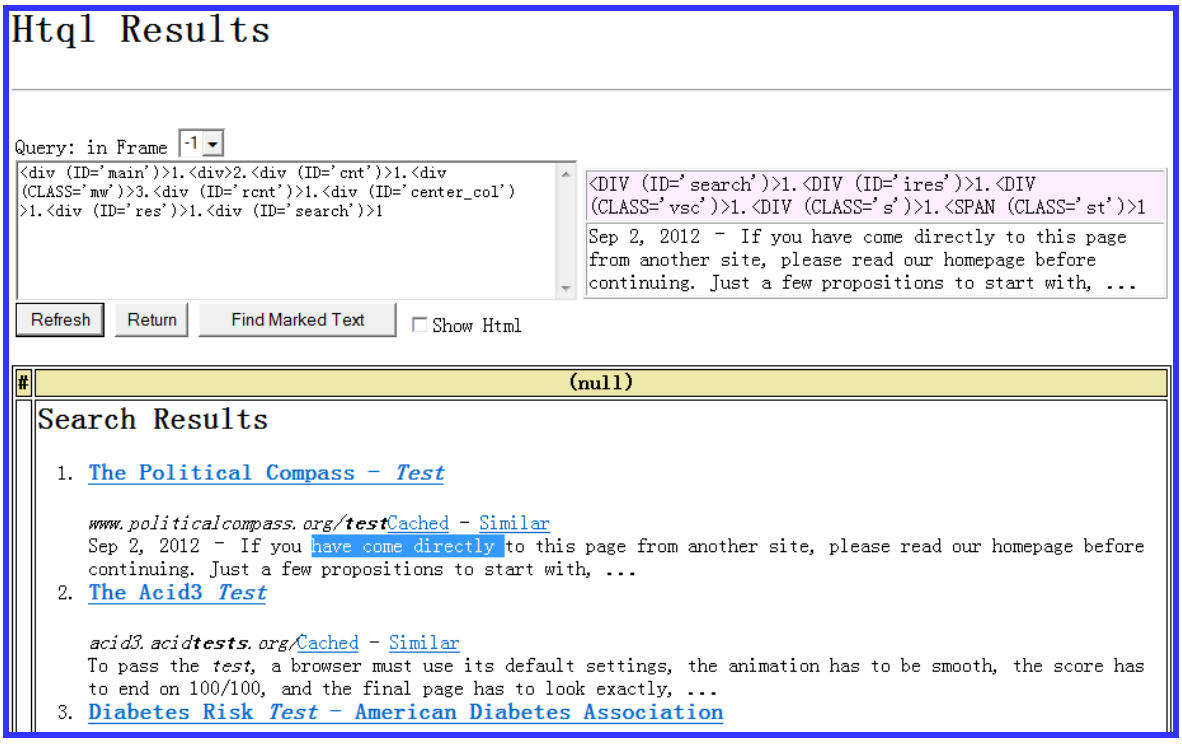
We try removing some trailing parts like:

<div (ID='main')>1 . 2 .<div (ID='cnt')>1 .<div (CLASS='mw')>3 .<div (ID='rcnt')>1 .<div (ID='center_col')>1 .<div (ID='res')>1 .<div (ID='search')>1 ~~.<span (CLASS='vshid')>2~~

and hit [Refresh], it shows:



This actually contains all the data we want. So we mark the "**have come directly**" again and hit [Find Marked Text] (figure below). It shows a query "<DIV (ID='search')>1.<DIV (ID='ires')>1.<DIV (CLASS='vsc')>1.<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1" on the upper right corner of the page. This is a query that can correctly locate the description text based on the existing extraction.

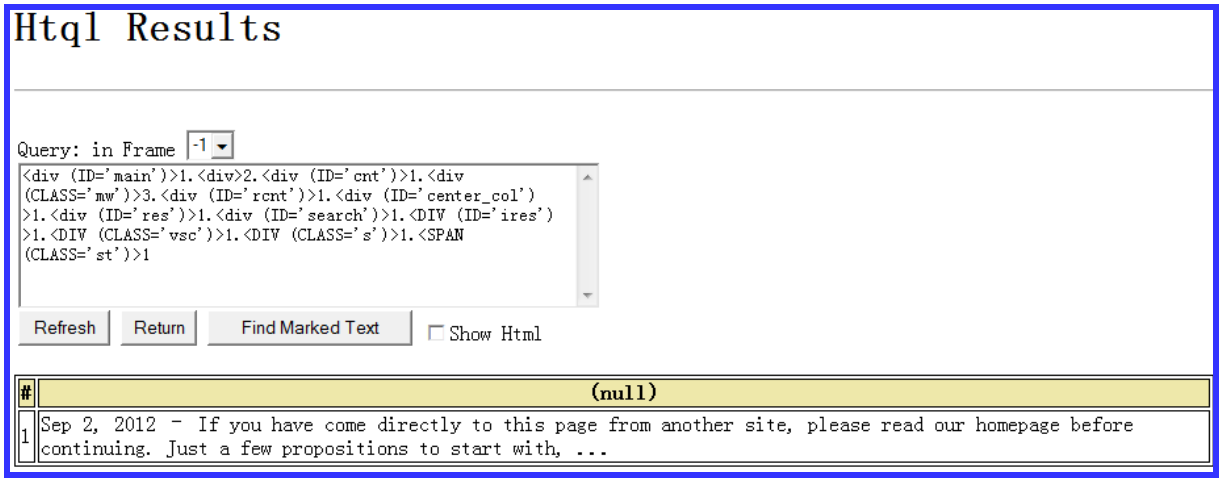


Now we can combine the new query with the original query, as in the following:

<div (ID='main')>1 . 2 .<div (ID='cnt')>1 .<div (CLASS='mw')>3 .<div (ID='rcnt')>1 .<div (ID='center_col')>1 .<div (ID='res')>1 **.<div (ID='search')>1** .<DIV (ID='search')>1.<DIV (ID='ires')>1.<DIV (CLASS='vsc')>1.<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1

Notice the "<DIV (ID='search')>1" part overlaps with the original query, and we should not enter it twice.

Hit [Refresh] and we get the description text we want:

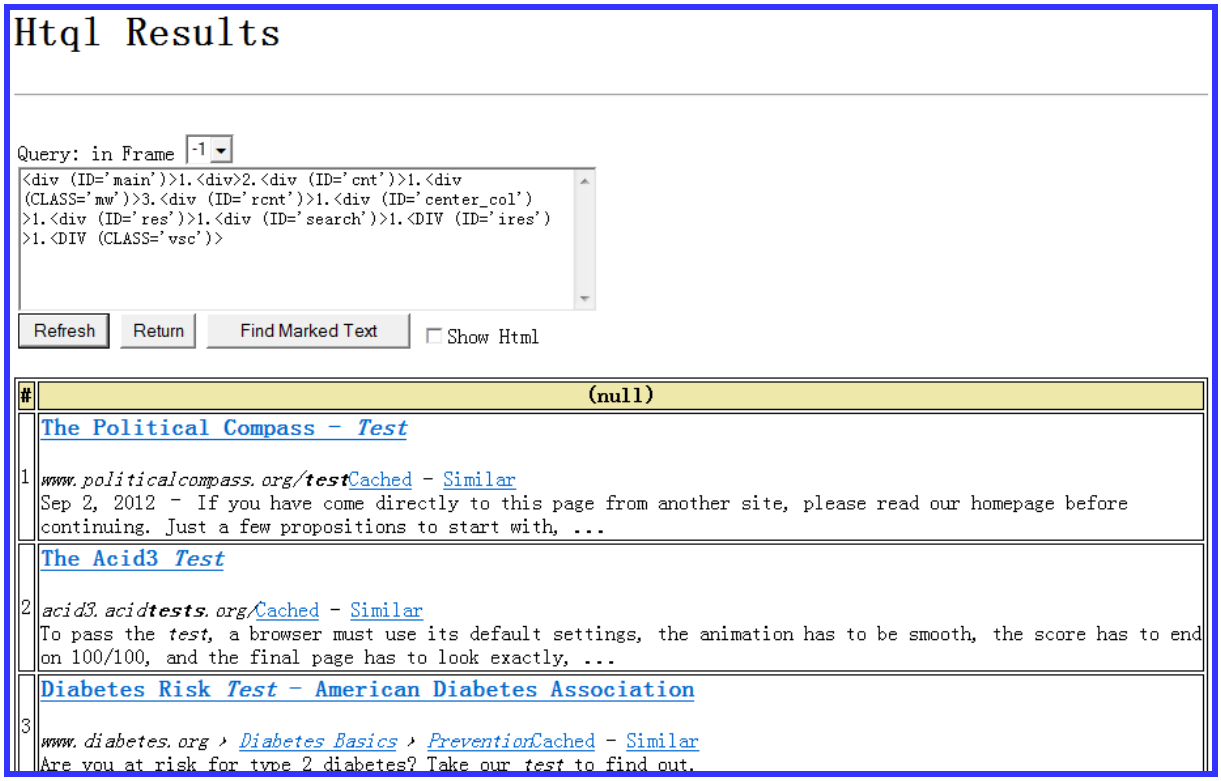


B) Compose Table Query

Now we want to extract all other descriptions as well. We know we will need a table query, but not sure how to construct one. We try by removing different trailing parts of the new query and see how it works. Specifically, we found that if we remove the trailing "1.<DIV (CLASS='vsc')>1.<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1", i.e., with query:

<div (ID='main')>1 . 2 .<div (ID='cnt')>1 .<div (CLASS='mw')>3 .<div (ID='rcnt')>1 .<div (ID='center_col')>1 .<div (ID='res')>1 **.<div (ID='search')>1** .<DIV (ID='search')>1.<DIV (ID='ires')>1.<DIV (CLASS='vsc')>~~1.<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1~~

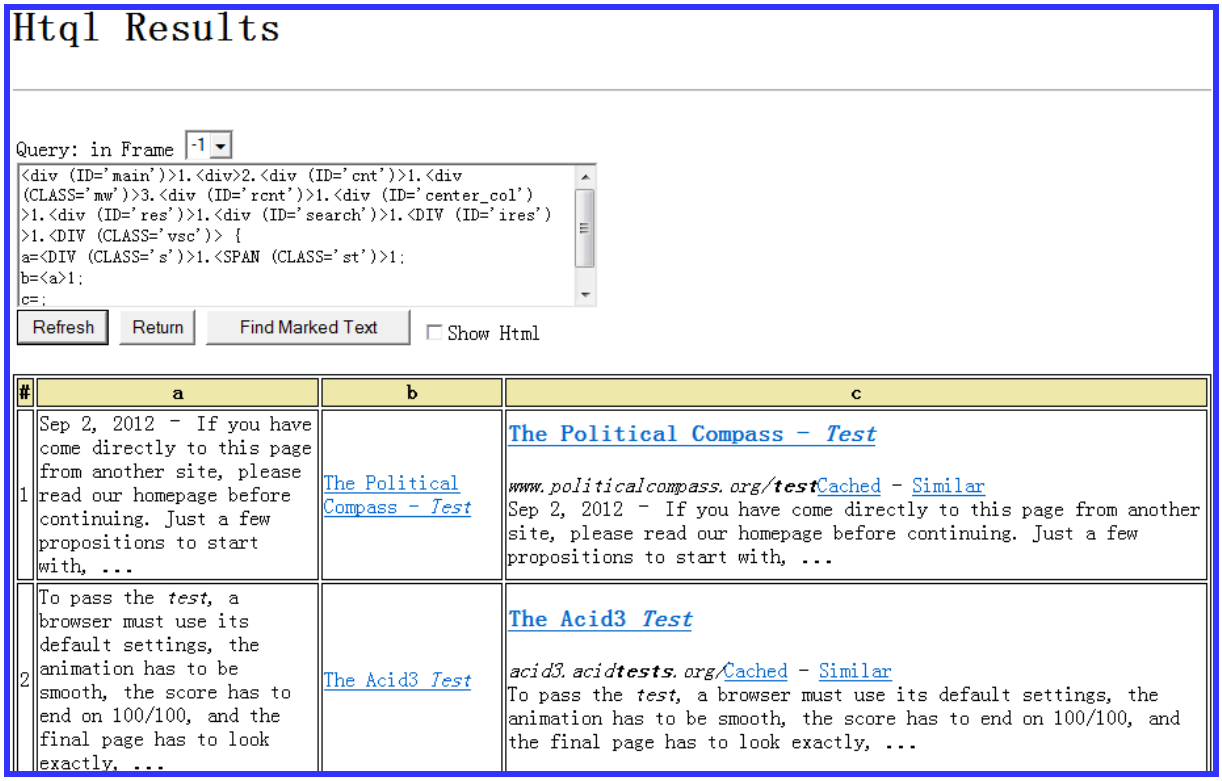
It extracts all the results we want.



We want to separate the results into a table. With a little HTQL syntax knowledge, we make this query:

```
<div (ID='main')>1 .<div>2 .<div (ID='cnt')>1 .<div (CLASS='mw')>3 .<div (ID='rcnt')>1
.<div (ID='center_col')>1 .<div (ID='res')>1 .<div (ID='search')>1 .<DIV
(ID='ires')>1.<DIV (CLASS='vsc')>
{
   a=<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1;
   b=<a>1;
   c=;
}
```

Notice we put back a part we just removed "<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1" into a column "a", set a column "b" to extract a link, and use column "c" to retain the entire item content. We get a perfect table:



We can use this query in a "Table a table" action to extract Google results.

C) Make It More Robust

In addition, we may want to simplify the query to make it more robust. For example:

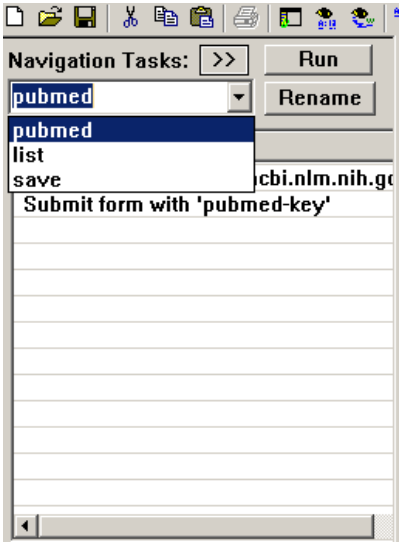
```
<div norecur (ID='search')>1 .<DIV (ID='ires')>1.<DIV (CLASS='vsc')>
{
   a=<DIV (CLASS='s')>1.<SPAN (CLASS='st')>1;
   b=<a>1;
   c=;
}
```

The "**norecur**" allows the query to drill down directly into a hierarchy of HTML div tags and find the target with the right ID. This query does not depend on outer structure and is more robust.

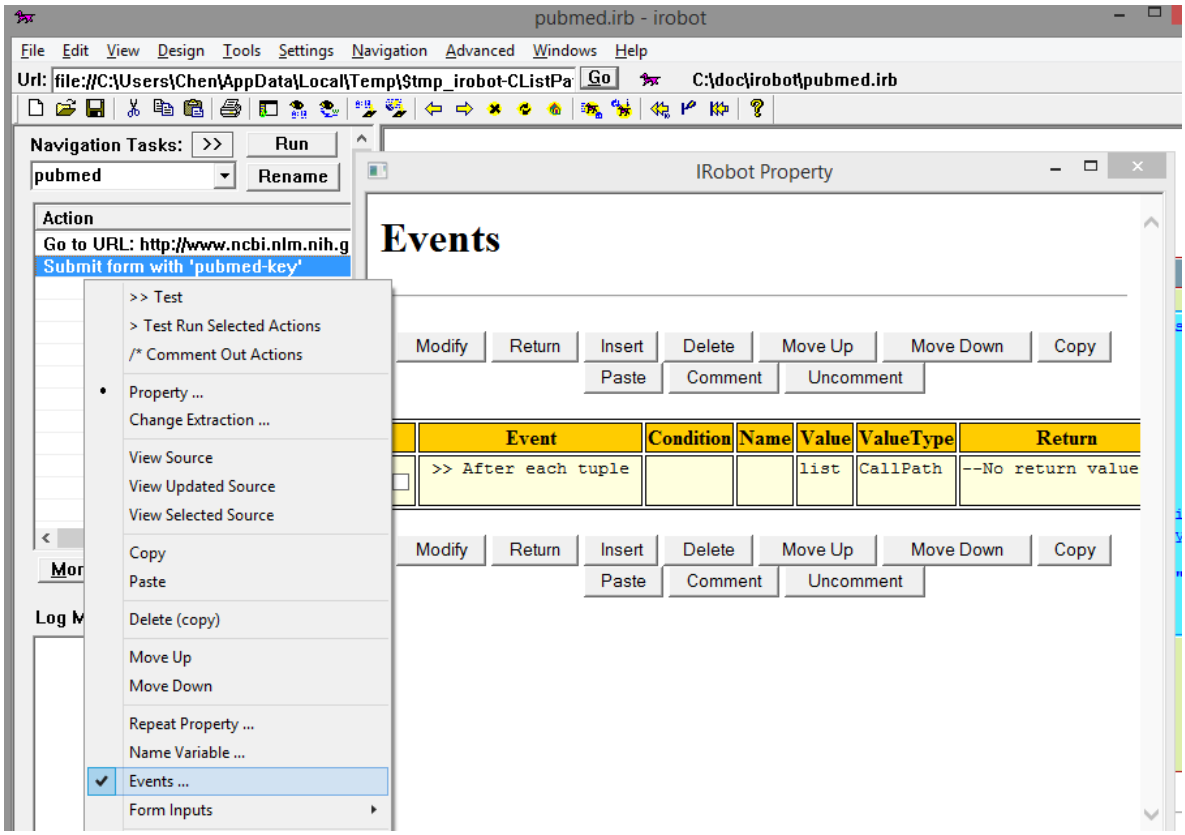
4. Managing Multiple Robot Tasks

4.1. Calling Robot Tasks

You may break down a complex robot into multiple robot tasks to modularize the robot program. These multiple tasks can then call each other recursively just like making functional calls in a regular program. The following figure shows that the sample robot "pubmed.irb" is divided into three tasks: "pubmed", "list", and "save".

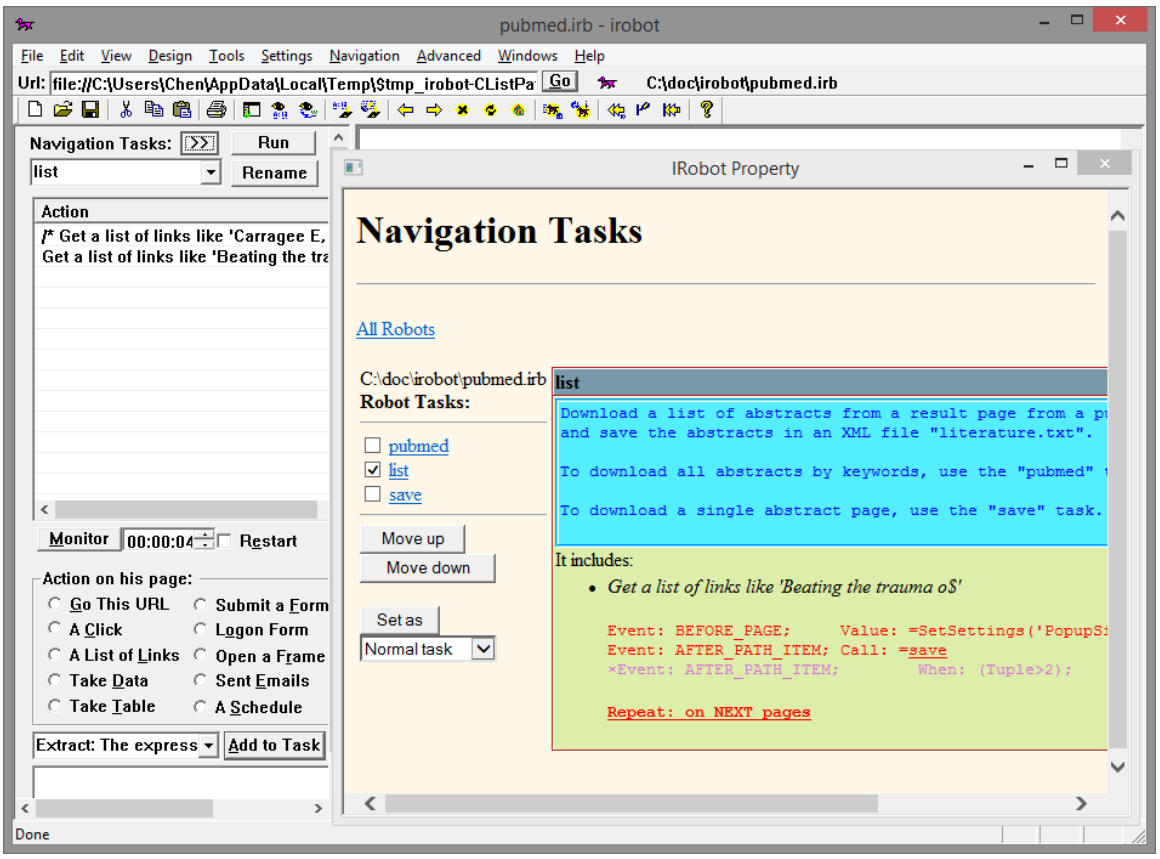


Robot tasks can be called from events. You can create an event by right-clicking on an action, selecting the "Events …" menu item, and inserting a new event. The following figure illustrate how the "pubmed" task calls the "list" task from an even of the "Submit form ***" action. The "CallPath" under the ValueType column specifies that the Value "list" is a task to call.



4.2. Organizing Robot Tasks

You may click on the [>>] after the "Navigation Tasks" to have an overview of the robot tasks. You can select a set of tasks to show a summary of them and move the relative position of robot tasks.



You may also set a task as "Runnable task", "Disabled task" or "Normal task" here. A runnable task will be listed when you left-click on the tray-bar icon, where you can run the task without showing the visual interface. Runnable tasks will also be shown in the main browser window as quick launches.

4.3. Parallelize Robot Tasks

When handling large volume data, it is idea that our robot can run tasks in parallel.

5. IRobot References

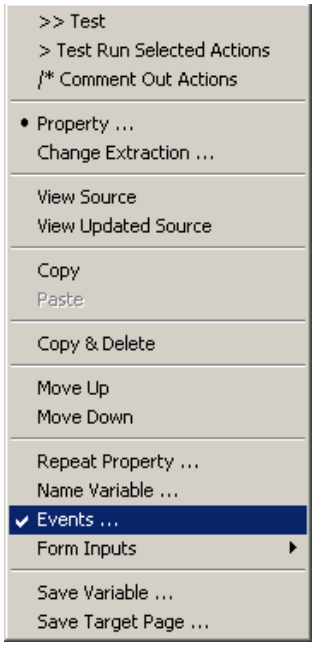
5.1. Event Definitions

Each robot action fires a number of events. Each event represents a time when the robot is opening, processing, or leaving a page. You can use events to create variables, compute expressions, and test conditions. The associated operations will then be carried out only at the designated time. You are able to create event rules by right-clicking on an action, and select the "Events…" The list of available events, in the order they are fired from the action, is:

a) Before ANY page: Before any page is open;

b) Before each page: Before the current page is open;
c) Read page: The page is just opened and HTML source is available;
d) Before each tuple: Before processing each tuple of the page;
e) Read tuple: Tuple variables are read from the page;
f) Before action: An action will be taken on the current tuple;
g) After action: Action on the current tuple is completed;
h) After each tuple: After each tuple is processed;
i) Completed each tuple: After the subsequent robot actions associated with the current tuple have been completed;
j) After each page: All tuple on the page has been processed;
k) After ALL pages: After all repeated actions are completed.

Action events are shown when right-clicking an action from the task action list, and choosing the "Events …" from the pop-up menu, as shown in the following figure.



The event page is shown in the browser window like:

Events

	Event	Condition	Name	Value	ValueType	Return
☐	Before action	ErrorCode>=0	a	ChangeFormInputs ('Please put login information in the browser and press OK.')	Expression	
☐	Before action	ErrorCode>=0 and a=-1			Expression	End this page
☐	After each tuple			Sleep(3)	Expression	--No return value--

Modify Return Insert Delete Move Up Move Down

Each event is defined by:

a) Event: The event when the associated expressions will be evaluated;
b) Condition: Evaluate the associated expressions only when the condition is satisfied;
c) Name: A variable will be created to contain the result of the Value expression;
d) Value: An expression for evaluation;
e) ValueType: Specify how the Value expression will be evaluated; it can be an expression, a string, a task call, or an HTQL expression against the current Web page (UpdatedPage);
f) Return: Some control flag or error code, for example, to end the execution on the current Web page.

5.2. Internal Variables

In addition to variables defined explicitly by users, there are a number of internally defined variables you can use directly.

TaskName: The current task the robot is running.

Tuple: The index of the current row of data in a page that the robot is acting on. When the robot completed all action, Tuple is reset to 0.

CurrentPage: For *repeat* actions, CurrentPage is the count of repeats on an action.

LastError: The status of the current action

a) LastError <0: the action is in error;
b) LastError =0: success;
c) LastError =1: ignore the current tuple and try the next tuple;
d) LastError =2: end the current action (and its subsequent actions) regardless of the rest tuples;
e) LastError =3: pause navigation on the current tuple;
f) LastError =4: retry on the current tuple.

ActionName: Take the current action name, each name corresponding to an action type. As a list:

a) Go to URL: ActionName='URL';
b) A Click: ActionName='Click';
c) A List of Links: ActionName='Table';
d) Take Data: ActionName='Item';
e) Take Table: ActionName='Table';
f) Submit a Form: ActionName='Form';
g) Logon Form: ActionName='Logon';
h) Open a Frame: ActionName='Frame';
i) Sent Email: ActionName='Email';
j) A Schedule: ActionName='Schedule'.

SourcePage: The original HTML source of the current page.

SourceUrl: The original URL of the current page.

UpdatedPage: The updated HTML source (by JavaScript) of the current page.

UpdatedUrl: The updated URL (by URL forwarding) of the current page.

TargetPage: The updated HTML source of the page currently in the browser.
TargetUrl: The updated URL of the page currently in the browser.
IrbFile: Full path name of the current irb file.
IrbFileName: Current irb file name without path.

5.3. Dataset Variables and Functions

A) Dataset Variables

A dataset variable is a set of data organized like a table in a database. You may load a dataset from a database or any string. The dataset can be set in the repeat property to loop over all tuples, or you can manually loop through all tuples. Dataset fields can be accessed like a regular variable.

We will demonstrate this functionality by showing how to read URLs from a text file and repeatedly navigate through all URLs. Suppose we have a text file "test.txt" with data (the separator can also be a TAB, ':', ';' or ','):

```
aaa|111|thttp://msn.com
bbb|222|http://yahoo.com
ccc|333|http://google.com
```

We are going to repeat a 'Go to URL' action for all the URLs in this text file. The steps are:

1) Create a 'Go This URL' action using any target page;

2) Insert a new event for this action: Event=Before any pages; Name=tuple_count; Value=newdataset.loadData('test.txt', '', 'txt');

This loads data from file 'test.txt' to a new dataset variable 'newdataset', and return the total number of tuples in the tuple_count.

3) In the "Repeat Property ..." of this action, check the "Repeat this action" checkbox, select the "Dataset" radio button, and enter "newdataset" below the ratio button;

This allows the robot to repeat on every tuple of the "newdataset".

4) In the "Property..." of this action, change the Url: to "newdataset.column3", and select the "Variable" after it.

This allows the robot to use URLs in the dataset for navigation. Dataset fields like "newdataset.column3" can be used anywhere like a regular variable. Default field names are column1, column2, ... for text files. For CSV, XML, and Access files, actual field names in the database will be used.

Dataset functions will be introduced in the following subsection.

B) Dataset Functions

Member functions for a dataset variable include:

1. loadData(dataset_location, [sql_query], [dataset_type], [htql_query])
2. addData(dataset_location, [sql_query], [dataset_type], [htql_query])

Load/add data from a data source. The dataset will be emptied first in the loadData() function.

The dataset_location can be any text file, CSV file, XML file, Access file, or a predefined data source.

The sql_query is an expression for querying the database.

The dataset_type can be 'txt', 'csv', 'xml', 'access', 'str', 'htql', 'var', or empty:

- *For dataset_types 'txt', 'csv', 'xml', and 'access', the dataset_location is the file location.*
  - Example: a.loadData('test.txt', '', 'txt');

- *For dataset_type 'str', the dataset_location is the actual string, and the htql_query is the field separator, such as a comma ','.*
  - Example: a.loadData('aaa,bbb,ccc', '', 'str', ',')
    Dataset a will include: 'aaa', 'bbb', and 'ccc'.

- *For dataset_type 'htql', the dataset_location is the actual string, and the htql_query is an HTQL query.*
  - Example: a.loadData('aaa,bbb,ccc','', 'htql', '/'','/')
    Dataset a will include 'aaa', 'bbb', and 'ccc'.

- *For dataset_type 'var', the dataset_location is the name of another dataset variable.*
  - Example: a.loadData('b', '', 'var')
    Dataset a will copy all tuples from dataset b.

- *For empty dataset_type, the dataset_location is a data-source name, and the sql_query is the SQL query that can be executed directly in the data source. In this case, the SQL query can be delete or update statements.*
  - Example: a.loadData('newdatasource', 'select * from newdatasource')
    Load all tuples in the newdatasource to dataset a.

  - Example: a.loadData('newdatasource', 'delete from newdatasource where ID=1')
    Delete a tuple in the newdatasource.

The htql_query is an expression for querying the source file or source database. If htql_query is given, the sql_query will be the SQL query to further filter tuples from the dataset (where the "FROM TABLE" clause can be omitted).

2. nameFields(fields)

Rename dataset fields.
The 'fields' is a list of comma separated field names.

3. next(), prev()

Move the dataset to the next/previous tuple. Note that in the above example, the next() is called implicitly by the repeat property.

4. first(), last()

Move the dataset to the first/last tuple.

5. query(sql_query, tuplejoint, fieldjoint)
6. query(sql_query, tuplejoint, fieldjoint)

Query the dataset with sql_query. If tuplejoint is not set, only the first result tuple will be returned. If fieldjoint is not set, only the first field will be returned. Results are concatenated together as a string using tuplejoint to joint tuples, and fieldjoint to joint fields.

7. sql(query)

Filter the dataset with a query. Tuples not satisfying the condition are deleted from the dataset.

8. formatHtml()

Format data in the dataset as an HTML table.

9. EOF()

Test if the dataset is ended.

10. MatchPatterns(string, [separator='',] [jointor=',',] [casesensitive='false',] [ds_is_pattern='false',] [string_is_pattern='false',] [from_tuple='str/tuple',] [max_matches=0]);

Match patterns in the tuples or in the *string*. Only the first field of the tuple is used. The function searches tuples that match any pattern in the *string*. Multiple patterns in *string* are separated by *separator*. Matching results are jointed by *jointor*. Matching results are taken from tuple if *from_tuple* = 'tuple', or from *string* if *from_tuple* <> 'tuple'.

C) Use Dataset Variables for Form Submission

Once a dataset variable is defined, its fields can be used like a regular variable. A typical use is for form submission. For example, in the above example, if we would like to submit the newdataset.column1, i.e., values of "aaa", "bbb", and "ccc" to google search repeatedly, then we need to design the robot as follows:

1) Create a 'Go This URL' action to go to http://google.com/;

2) Insert a new event for this action: Event=Before any pages; Name=tuple_count; Value=newdataset.loadData('test.txt', '', 'txt');

This loads data from file 'test.txt' to a new dataset variable 'newdataset', and return the total number of tuples in the tuple_count.

3) In the "Repeat Property ..." of this action, check the "Repeat this action" checkbox, select the "Repeat" radio button, and enter "newdataset" after the ratio button;

This allows the robot to repeat on every tuple of the "newdataset".

4) Add a submit form action to submit the google form.

5) Double click the form action, you will be in the "Form Values" panel. Find the input corresponding to the search term, which can be identified by the sample text you input to the search box, right-click on it, and select "From Variable…". Then input the "newdataset.column1" (without quotation marks) in the popup box.

5.4. Internal Functions

A) String Functions

Str ~ RegEx

Search from *Str* a substring with a regular expression *RegEx*. For example, 'abc123' ~ '\d+' would get '123'.

You can use most of the common regular expression syntax such as \w for any alphabet character, \d for digit, \s for white space, and \W, \D, and \S their negations. In addition, you can use expression like '&[s:datasetvar]' to match any string in a dataset (only the first column is used). For example: you can load a list of countries into a data variable, and then search the string for possible matches:

Event: Before any page; Value: countries.loadData('countries.txt')
Event: After each tuple; Name: p; Value: 'I live in United States' ~ 'live in &[s:countries]'

Str ~= RegEx

Match *Str* with a regular expression *RegEx*. For example, 'abc123' ~= '\d+' would get 0, and 'abc123' ~= '\w+\d+' would get 1.

NewString=substr(String, StartPosition0, [Len])

Get the substring of a string from StartPosition0 of length Len.

Pos0=strfind(String, Substring, [StartPos0])

Find the first occurrence of Substring in String, starting from StartPos0. If no occurrence is found, return –1. If *StartPos0<=0*, search from the first position.

NewString=strcat(String1, String2, …)

Concatenate multiple strings into a single string.

Len=strlen(String)

Return the length of a string.

NewString=replace(String, Source, Target)

Replace any occurrence of Source in String to Target.

NewString=ltrim(String, TrimedChars)

Delete from the left of *String* any occurrence of chars in *TrimedChars*.

NewString=rtrim(String, TrimedChars)

Delete from the right of *String* any occurrence of chars in *TrimedChars*.

FormattedString = format(fmt, ...)

Format a string. Refer *printf* function in C.

Result = htql(TextSource, HtqlQuery)

Evaluate an HTQL expression against data in TextSource. Only the first result is returned.

Result = get_emails(Text, fmt)

Find emails from *Text*. If *fmt* is set, *Results* are formatted in *fmt*; otherwise, Results are formatted as: "name" <email>;. In *fmt*, %m will be replaced by email, %n will be replaced by name, multiple emails will be concatenated after formatted.

B) Date Time Functions

TimeInt=time()

Return the current time in integer.

DateVar=date(), sysdate()

Return the current date.

NewNumber=to_number(String)

Convert a string into a number.

StrDate=to_char(IntDate, DateFormat)

Return a string representation of a date in a certain format.

IntDate=to_date(StrDate, DateFormat)

Return an integer date from a date string in a certain format.

C) Floating Number Functions

RandNum=rand([RandMax])

Return a random number less than RandMax; If RandMax>1, RandNum is integer, if RandMax=1, RandNum is a floating number.

IntNum=round(FloatNum)

Return the rounded integer number of a floating number.

IntNum=floor(FloatNum)

Return the truncated integer number of a floating number.

IntNum=ceil(FloatNum)

Return the next integer number of a floating number.

FloatingNum=get_number(text, index)

Find a *index*'th number from the *text*.

D) Data Type Functions

Test=ischar(var)

Test if *var* starts with an alphabet character, except any leading spaces.

Test=isnumber(var)

Test if *var* starts with a digit of '0'-'9', except any leading spaces.

Test=isblank(var)

Test if *var* is a composed of blank characters.

Test= isphonenumber(var)

Test if *var* is a valid phone number, i.e., as a combination of '0'-'9', '+', '-', '(', ')', and

'.', and has a length of 6-20 characters.

E) File Functions

SaveUrlFile(URL, FileName)
Download a file at *URL* and save to a file *FileName*.

Data=UrlData(URL)
Get data from *URL*. URL can be a local or http file.

Data=ReadFile(FileName)
Load data from local *FileName*.

SaveFile(FileName, DataString)
Save *DataString* to a file of *FileName*.

AppendFile(FileName, DataString)
Append *DataString* to file *FileName*.

DeleteFile(FileName)
Delete file *FileName*.

CapturePage(*ImageFilename*)
Capture the current web page and save it as an image. Only BMP file format is supported now.

Name = GetFileName(FileName)
Get the file name of *FileName*.

Path = GetFilePath(FileName)
Get the path of *FileName*.

MkDir(DirName)
Create a directory named *DirName*.

UniqueFileName = GetUniqueFileName(Dir, FileName, [SubDir])
Get a unique file name in *Dir*, or *Dir*/*SubDir*/, with the base name given in *FileName*. If *Dir* or *Dir*/*SubDir*/ does not exist, the directory will be created by this function. The returned *UniqueFileName* does not include the *Dir* prefix, but includes the *SubDir* prefix. For example: *GetUniqueFileName('data', 'test.html', '11-05-01')* created a *'data/11-05-01/'* directory and returns *'11-05-01/test.html'* .

List = DirFiles(*Match*, 'FILE | DIR | RECUR | PATH | NAME | PATHNAME | WRTIME |

CRTIME | SIZE')

Get file information. *Match* is a filename, may include wildcards * and ?.

'FILE|DIR': search filename or dir name or both.

'RECUR': search subfolders.

'NAME|PATH|PATHNAME|SIZE|WRTIME|CRTIME': return the list of information for each file, separated by the same separator (e.g., '|' here) each line.

SendEmail(*email_data*, *to_email*)

Send *email_data* to a recipient email *to_email*.

F) Interface Functions

NewValue=InputText(Description, DefaultValue)

Prompt user for input.

Count=InputVariables(Title, Name1, Value1, Desc1, Size1, Name2, Value2, Desc2, Size2 ...)

Prompt user for setting of multiple variables at the same time.

Title: title of the input dialog;

Name: variable name;

Value: default variable value;

Desc: description of the variable;

Size: size of the input box if it is a number, or formatted selection options, e.g.:

|SELECT|option1:value1|option2:value2

|RADIO|option1:value1|option2:value2

|CHECKBOX|option1:value1|option2:value2

Sleep(Seconds1, Seconds2)

Sleep for a certain amount of time in seconds; If Seconds1<0, then sleep for a random amount of time less than the absolute value of Seconds1; If both Seconds1 and Seconds2 <0, then sleep for a random amount of time in between the absolute values of Seconds1 and Seconds2.

CallPath(TaskName, IrbFile, Passwd, Session)

Call *TaskName* task in robot file *IrbFile* with password *Passwd*. If *Session* is specified, the new task will run in a separate window named *Session*.

CallParallel(*TaskName*)

Call *TaskName* task in a parallel thread. The parallel thread only uses HTTP socket browser, and will not execute JavaScripts.

ShowMessage (*Message, WaitSeconds*)

Show *Message.* If *WaitSeconds* is given, close the meesage after *WaitSeconds* time.

PathAlertMessage (*Message*, *LinkTask*)

Add an alert *Message* in the taskbar icon, and play a sound. If *LinkTask* is a task name, the alert *Message* is linked to the *LinkTask* when clicked, otherwise, it works as a message ID. Typical use: *PathAlertMessage* (*Message*, *time(0)*).

G) Browser Functions

GetCookie(CookieName)

Get the cookie associated with a CookieName.

SetCookie(CookieName, CookieValue)

Set a cookie in the current page.

ClearCookies([Url])

Delete cookies associated with *Url*. If *Url* is not given, delete the cookies associated with the current URL and URLs from the same domain. If *Url*=='*', then delete all cookies in the system.

SetProxyServer([IP], [Port], [SessionAgentName])

Set the proxy server with *IP* and *Port*. Remove proxy setting if no parameters are given. If *SessionAgentName* is not provided, the proxy setting will affect system-wise user settings. Otherwise, it will serve as the agent name of the current session, and the proxy setting will affect this session only. Use, for example, "Microsoft Internet Explorer" for the *SessionAgentName*.

SetAttribute(Htql, AttributeName, AttributeValue)

Modify HTML attribute in the current page, *Htql* specifies the tag for modification. *AttributeName* can be any attribute in the tag, or 'innerText', 'innerHtml', 'outerText', and 'outerHtml'.

SetInput(InputName, InputValue)

Set input data in the current page.

FireEvent(ItemHtql, EventName)

Fire HTML event for the target item.

JavaScript(statement)

Run JavaScript *statement*. The statement can be jQuery as well. An example to trigger a change in jQuery is:

```
JavaScript('jQuery("#input-id" ).trigger("change") ')
```

ScrollTo(x, y)

Scroll browser to (*x*,*y*) position.

MouseMove(ItemHtql, [offset_x, offset_y])
Move mouse over the item specified by *ItemHtql*, with an offset of *offset_x, offset_y.*

MouseClick(ItemHtql)
Simulate a mouse click at *ItemHtql*, or at the current mouse position.

MouseRightClick (ItemHtql)
Simulate a mouse right click at *ItemHtql*, or at the current mouse position.

Refresh()
Refresh the current page.

GoForward()
Navigate forward.

GoBack()
Navigate backward.

ReloadPage('URL/LOCAL/');
Reload page.

ClickItem(*item_htql*, *MaxWaitSec*)
Click a page item specified by *item_htql*.

GoUrl(*url*)
Go to URL *url*.

LoadPage(*page_html*)
Load *page_html* to the browser.

H) Automation Functions

Titles = GetWindows([Title])
Get application window titles. If *Title* is given, it finds the best matched title from all application titles. Otherwise, it gets all application window titles and separates them with a new line.

RunApp(CommandLine) or
StartApp(CommandLine)
Run application using *CommandLine*.

SendAppText(ApplicationTitle, TextString)

Find the application with title *ApplicationTitle*, and send text *TextString* to it.

SendAppKeys(ApplicationTitle, KeyString)

Find the application with title *ApplicationTitle*, and send keys *KeyString* to it. Refer *SendKeys* in .NET. The plus sign (+), caret (^), percent sign (%), tilde (~), and parentheses () have special meanings to SendKeys. To specify one of these characters, enclose it within braces ({}). For example, to specify the plus sign, use "{+}". To specify brace characters, use "{{}" and "{}}". Key Code: SHIFT +; CTRL ^; ALT %; BACKSPACE {BACKSPACE}, {BS}, or {BKSP}; BREAK {BREAK}; CAPS LOCK {CAPSLOCK}; DEL or DELETE {DELETE} or {DEL}; DOWN ARROW {DOWN}; END {END}; ENTER {ENTER} or ~; ESC {ESC}; HELP {HELP}; HOME {HOME}; INS or INSERT {INSERT} or {INS}; LEFT ARROW {LEFT}; NUM LOCK {NUMLOCK}; PAGE DOWN {PGDN}; PAGE UP {PGUP}; PRINT SCREEN {PRTSC}; RIGHT ARROW {RIGHT}; SCROLL LOCK {SCROLLLOCK}; TAB {TAB}; UP ARROW {UP}; F1 {F1}; F2 {F2}; Keypad add {ADD}; Keypad subtract {SUBTRACT}; Keypad multiply {MULTIPLY}; Keypad divide {DIVIDE}.

SendAppTextWait(TimeSec, TextString, WindowTitle, ChildWinTitle, Delay)
SendAppKeysWait(TimeSec, KeyString, WindowTitle, ChildWinTitle, Delay)

Find the application with window title *WindowTitle*, and send keys *KeyString* to it's child window whose title is *ChildWinTitle*. Wait for *TimeSec* in the background if the application or the child window cannot be found. If *TimeSec* < 0, continue robot actions while waiting; otherwise, wait until the window is found. After the application is found, wait for *Delay/3* seconds before sending the *KeyString* or *TextString*. For example, use the following function with a "Before each tuple" event for a "Click" action to a file button to upload a file.

SendAppTextWait(-60, filename+'\n', 'Choose File to Upload', '', 5)

CloseApp(ApplicationTitle)

Close an application.

Copy(text)

Copy *text* to the clipboard.

Text = Paste()

Get *Text* from the clipboard.

I) Browser Proxy Functions

These functions are specific to browser proxy use, such as from htql.Browser(2). (See HTQL Python manual at http://htql.net/)

Html = GetPage()

Get page source HTML.

Url = GetUrl()

Get page source Url.

Html = GetUpdatedPage()

Get page updated source from the current frame.

Url = GetUpdatedUrl()

Get page updated Url from the current frame.

GoUrl(url, wait_sec)

Go to Url url, wait for maximum of wait_sec.

Click(item_htql, wait_sec)

Click on item defined by *item_htql*, wait for maximum of wait_sec.

UseFrame(frame_index[, frame_htql])

Use the *frame_index* or *frame_htql* as the current frame.

UseForm (form_htql)

Use the *form_htql* as the current form.

SetFormValue (input_name, input_value)

Set the *input_name* of the current form to value *input_value*.

SubmitForm (form_htql, input_htql, wait_sec)

Submit the form defined by *form_htql*, with submit button at *input_htql*, and wait for a maximum of *wait_sec* time.

SetVariable (name, value)

Set variable *name* to be *value*.

Value = GetVariable (name)

Get variable *name*'s value.

OpenRobot (irbfile, irbtask, irbpass)

Open the robot of *irbfile*, set *irbtask* to be the current task, and use *irbpass* as the passcode.

RunTask(irbtask, flag)

Run the task *irbtask* of the current robot. If *flag*== 'keepvar', then robot variables will not be cleared, otherwise, all robot variables will be cleared before running. If

flag==’continue’, then it will keep all the robot status and continue the execution from task.

Status = IsStopped()

Return ‘YES’ if the robot is not running, or ‘NO’ otherwise.

J) Crawling Functions

SetDownloadParameter(threads, max_queue)

Set the number of parallel threads and the size of waiting queue.

SetDownloadCompletionTask (Id, CallTask)

Call *CallTask* after crawler Id completed.

Id = SearchEmail (StartUrl, EmailFile_CallTask, MatchPatterns, ExcludePatterns, MaxDepth)

Search emails from StartUrl and its subdirectories. If *EmailFile_CallTask* is a task name, call the task on each crawled Web page, otherwise, save emails to the file named *EmailFile_CallTask*. if *MatchPatterns* is given, only URLs including the pattern will be crawled; if *ExcludePatterns* is given, any URLs including the pattern will be excluded; if *MaxDepth* is given, the crawler will only go *MaxDepth* steps from StartUrl. This function only works in the paid version.

Id = SearchSiteLinks (SourceSites, TargetSites, EmailFile_CallTask, MatchPatterns, ExcludePatterns, MaxDepth)

Search links from *SourceSites* to *TargetSites*. Urls in *SourceSites* to *TargetSites* are separated by new lines (\n). See *SearchEmail()* function for other parameters.

Id = CrawlWebsite (SourceSites, CallTask, MatchPatterns, ExcludePatterns, MaxDepth, targets)

Search links in *SourceSites*, and call CallTask on each crawled page. Urls in *SourceSites* are separated by new lines (\n).

targets is the types of target links to crawl. It can be a combination of ‘A’ (href links), ‘FRAME’ (frame pages), ‘FORM’ (form targets), ‘IMG’ (image files), ‘SCRIPT’ (js files), and ‘LINK’ (css files) separated by ‘|’. The default value is ‘A|FRAME|FORM|IMG’.

See *SearchEmail()* function for other parameters.

Id = DownloadPage (SourceSite, SaveDir, MatchPatterns, ExcludePatterns, MaxDepth)

Crawl *SourceSite*, and save all files in *SaveDir*. See *SearchEmail()* function for other parameters.

K) Setting Functions

SetProgram ('Window', 'Show/Hide/Maximize/Minimize')
Show IRobot program.

SetProgram ('Program', 'Exit')
Quit IRobot program.

SetProgram ('ToolPanel', 'Show/Hide/Toggle')
Toggle the tool panel.

SetSettings ('Speed', 'Fast/SuperFast/Slow/VerySlow/NonStop')
Set IRobot navigation speed.

SetSettings ('Popup', 'block/nonblock/InBrowser')
Set IRobot popup options.

SetSettings ('PopupSize', 'Large/Middle/Small')
Set IRobot popup window options.

SetSettings ('Silent, 'Yes/No/True/False')
Set IRobot silent browsing options.

5.5. Scripting

IRobot supports a subset of C language script. Scripts can be used in the Value field of events, when the ValueType is set as Script. Example code snippets:

if … else …	a='rain';
	if (a=='rain' or a=='cloud'){
	  c='';
	}else if (a=='sun') {
	  c=sysdate();
	}
	return c;

For …	a=10;
	for (i=0; i<3; i+=1) {
	  a=a-1;
	}
	ShowMessage(a, 3);

foreach …	vardataset.loadData('test.txt',','csv')
	options='|SELECT|';
	foreach (val=ValueField, url=Url: vardataset){
	  options=options+&val+&':'+&replace(url,':',':')+& '|';
	}
	InputVariables('Test','xxx',222,'Select this URL', options)
	return xxx;

do … while	a=10;
	do {
	  a-=1;
	} while (a>3);
	return a*2;

while …	n=10;
	while (n>0){
	  n-= 2;
	}
	return n;

function	function f1(b){
	  return b+f2(b);
	}
	function f2(b){
	  if (b==1) return 1;
	  return b*b+f1(b-1);
	}
	d=f1(100);

Notes:

- Global variables can be accessed in the entire robot
- Script functions are local and can only be accessed from the same script
- Function local variables can only be accessed from within the function
- Variable types are interpreted on the fly
- All IRobot functions can be used, but only one function can involve interface
- i++ and i-- are not supported, use i=i+1, or i+=1 instead
- The last statement or the "return" statement returns value to the Name field of the Event.

5.6. Command Line Parameters

Irobot.exe xxx.irb /run:taskname
irbrun.exe xxx.irb /run:taskname
Open robot *xxx.irb* and run the *taskname* task.

/exit
Quit irobot program after completed the task.

/yes
Automatically choose YES when irobot asks for confirmation

/no
Automatically choose NO when irobot asks for confirmation

/pass:passcode
Use passcode to open the robot

/maxtime:xxx
Run at most *xxx* seconds of time, *xxx* can be an expression.

/var:xxx=yyy
Set variable *xxx* to value *yyy*

5.7. Detailed Action Properties

1) Go to URL

The property page looks like:

The Url can be a String like 'http://mail.yahoo.com/' in the example. The Url can also

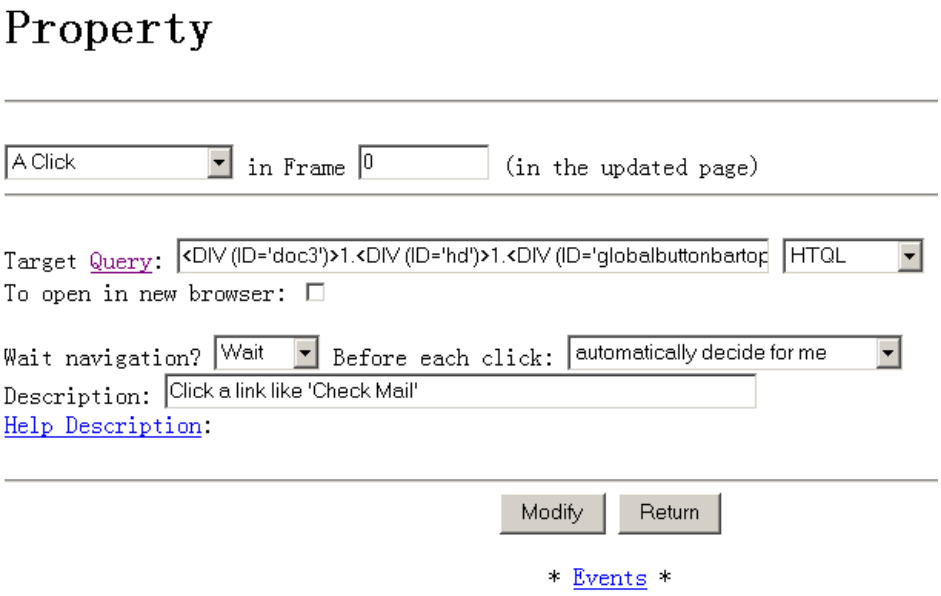
be taken from a Variable or an Expression, such as 'http://mail' + '.yahoo.com'.

Make sure to click the [Modify] button after any change.

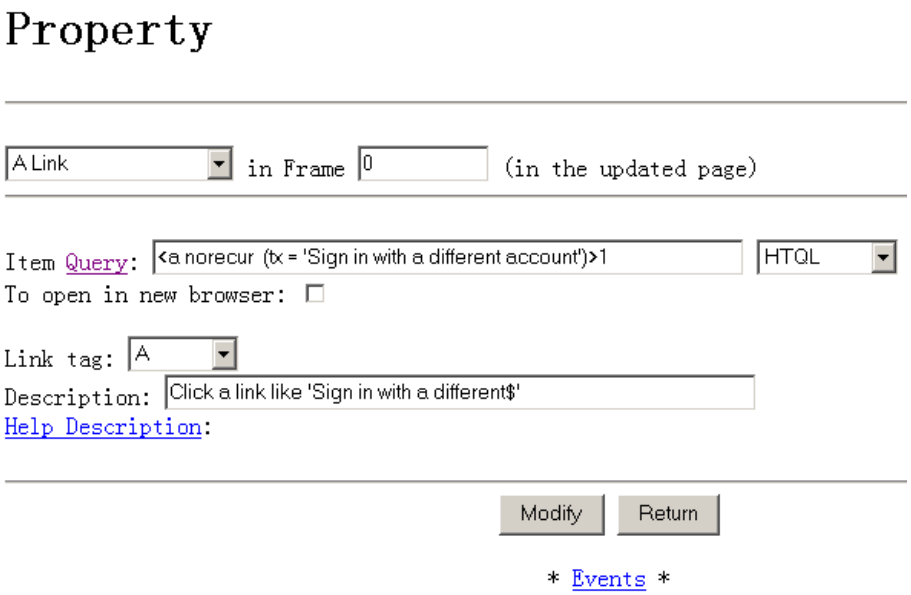
The "Description" will be shown in the action list when showing this action. You can use the Help Description to add memo or notes for this action.

2) A Click

The property page looks like:



"A Click" is similar to "A Link", which can be chosen from the drop-down list. "A Click" is more general because it can also click on a JavaScript link. The property page of "A Link" looks like:



The "in Frame" specifies which child frame the link is in. For the first child frame, it is 1, the second child 2, and so on. Frame 0 is the current web page including any content created by dynamic scripts. Frame -1 is to use the source code of the current web page without any script interpretation. If there are several layers of frames, use something like "2.1.3" to access "the second child frame, and the first child frame, and the third child frame."

The "Query" link tests if the target query locates the correct link. In order to use the test correctly, navigate to the target page before opening the property page.

"Target query" is an HTQL expression to locate the link to click. The HTQL expression can also taken from a Variable or an Expression.

If you check the "To open in new browser", the target page will be opened in a new browser.

"Wait navigation?" specifies whether to wait for navigation after the click or not.

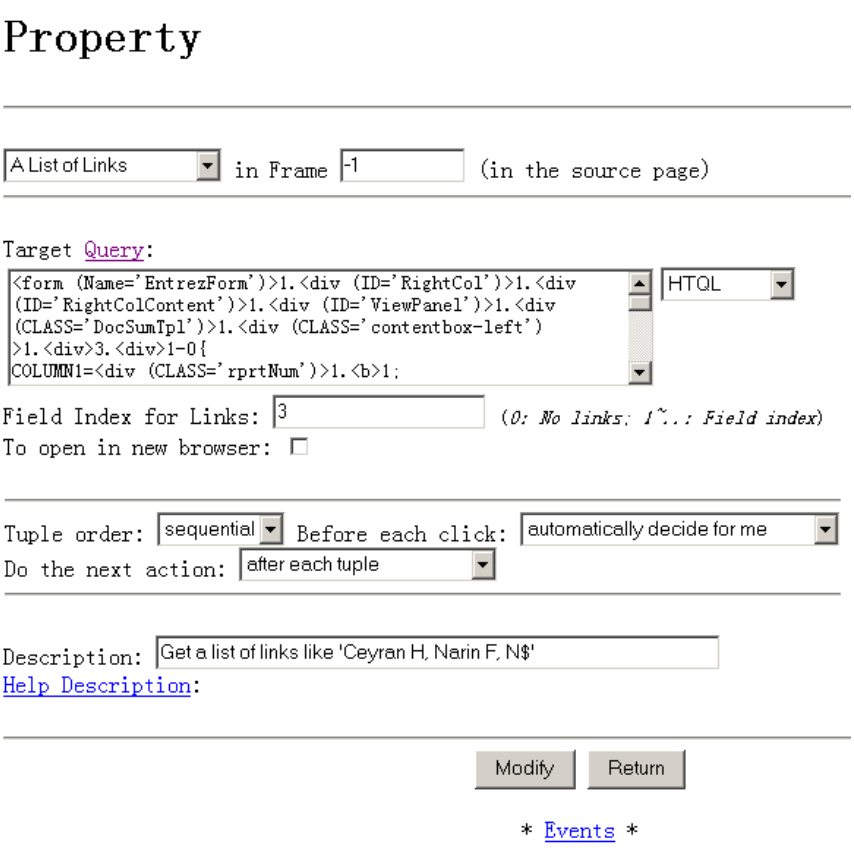
For the "Before each click", you can decide if you want to reload the web page for clicking or not. It is most useful when you use this in the "Repeat property" for another action, so that you need to bring up the page before clicking on the Next link. Refer "Repeat property" in section "Repeat on Next Pages" for more help.

The "Link tag" specifies the HTML tag of the target link.

Make sure to click the [Modify] button after any change.

3) A List of Links

The property page looks like:



The "in Frame" specifies which child frame the link is in. For the first child frame, it is 1, the second child 2, and so on. Frame 0 is the current web page including any content created by dynamic scripts. Frame -1 is to use the source code of the current web page without any script interpretation. If there are several layers of frames, use something like "2.1.3" to access "the second child frame, and the first child frame, and the third child frame."

The "Query" link test if the target query locates the correct list of links. In order to use the test correctly, navigate to the target page before opening the property page.

"Target query" is an HTQL expression to locate the list of links to click. The HTQL expression can also taken from a Variable or an Expression.

"Field Index for Links" specifies the table column where the links are in. If the field index is 0, the action is turned to 'Take Table', without following any links.

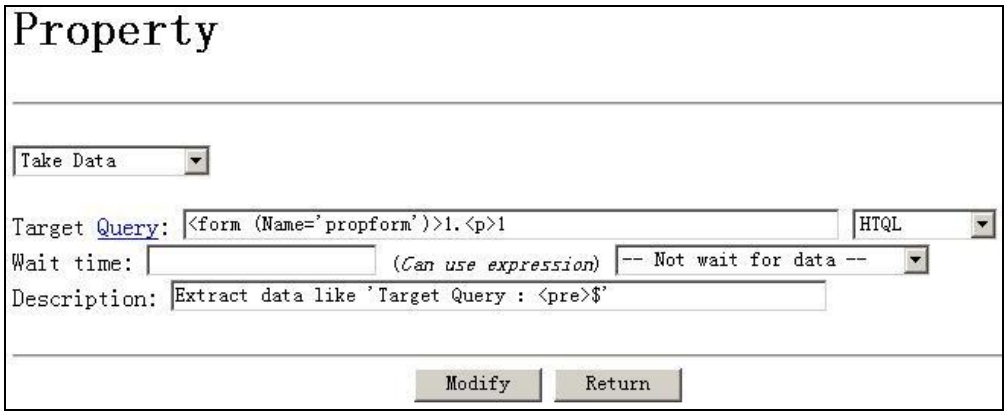
If you check the "To open in new browser", the target pages will be opened in a new browser.

"Tuple order" specifies if to open the link sequentially or reversely. The "Before each click" specifies whether to reload the web page for link clicking or not. If the "Do the next action" is "after completed this page", then it will click all links before doing the next action. Otherwise, it will repeatedly do the next action for each link.

Make sure to click the [Modify] button after any changes.

4) Take Data

The property page looks like:



The "Query" link tests if the target query extracts the correct data. In order to use the test correctly, navigate to the target page before opening the property page.

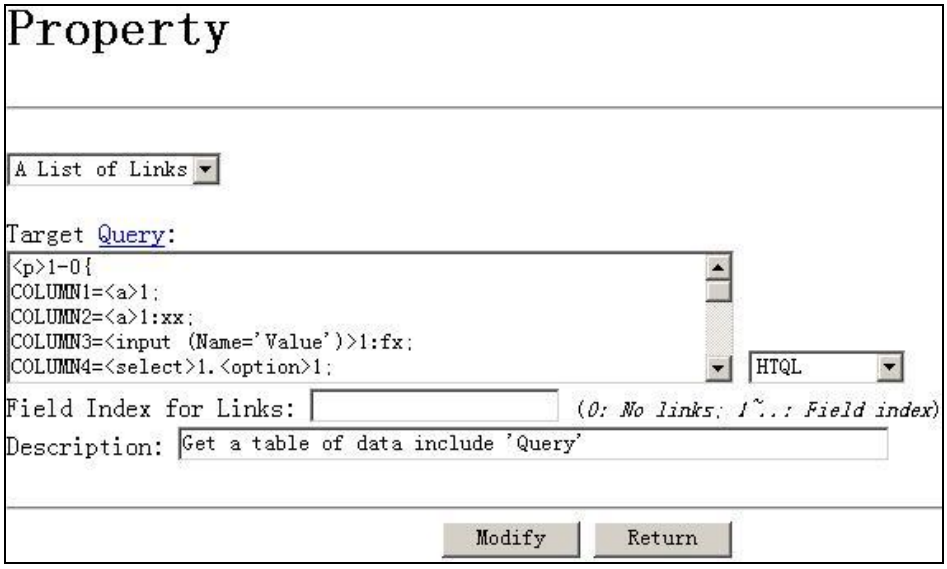
"Target query" is an HTQL expression to extract the data. The HTQL expression can also taken from a Variable or an Expression.

You can let the robot wait until the target data is shown or until the data disappear (work only when the page is refreshing by itself). "Wait time" specifies the wait interval in seconds.

Make sure to click the [Modify] button after any change.

5) Take Table

Take table is the same as "A List of Links", when the "Field Index for Links" is set to 0 or left empty. Example page:



The "Query" link tests if the target query locates the correct list of links. In order to use the test correctly, navigate to the target page before opening the property page.

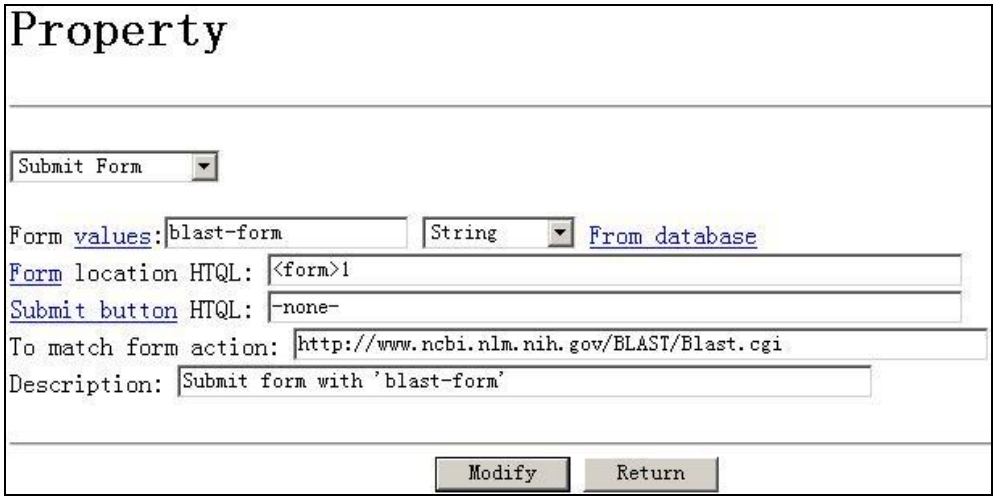
"Target query" is an HTQL expression to locate the list of links to click. The HTQL expression can also taken from a Variable or an Expression.

"Field Index for Links" specifies the table columns where the list of links is defined in the target query. If the field index is 0, the action is turned to 'Take Table'.

Make sure to click the [Modify] button after any change.

6) Submit a Form

The property page looks like:



The "values" link brings up the form value window from the control panel (Click on the [OK] button in the form value window to close it).

"Form values" specifies the form values to be filled in the target form. The form values can be specified as a "Variable" or an "Expression". Form values can be drawn from databases, which will be explained later.

The "Form" link tests if the target form is located correctly. In order to use the test correctly, navigate to the target page before opening the property page.

"Form location HTQL" is an HTQL expression to locate the form to be submitted.

"Submit button HTQL" is an HTQL expression to locate the submit-button of the form. A special '-none-' expression tell the robot not to submit the form (only fill form values).

"To match form action" tells the robot submit the form only the action matches the specified URL.

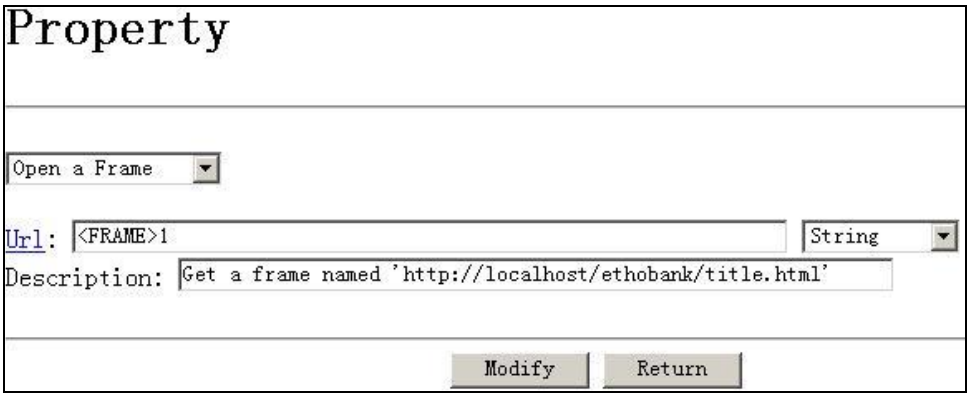
Make sure to click the [Modify] button after any change.

7) Logon Form

'Logon Form' is same as 'Submit a Form'

8) Open a Frame

The property page looks like:



The Url specifies an HTQL expression to locate the frame. The Url can also be taken from a Variable or an Expression, such as 'http://mail' + '.yahoo.com'.

Make sure to click the [Modify] button after any change.

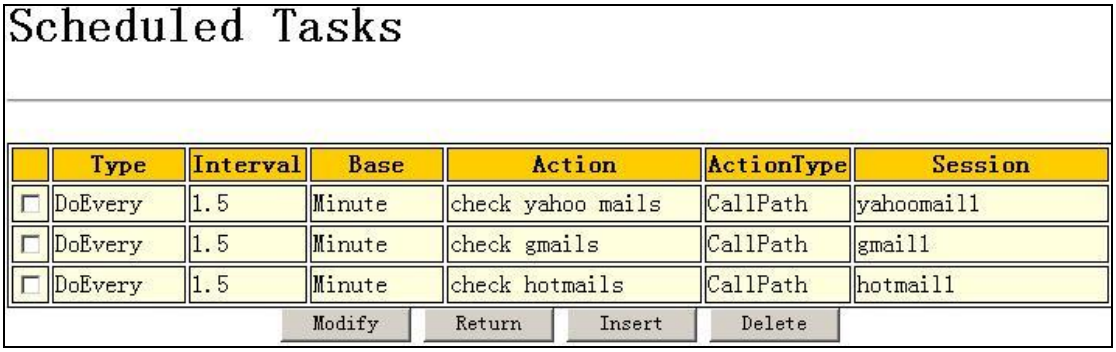
9) Sent Email

To be explained.

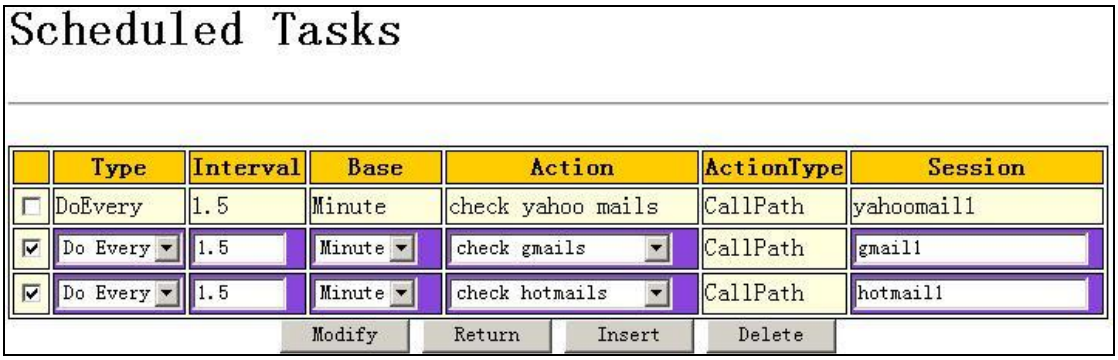
10) A Schedule

The property page looks like:

Click on the "View Schedule" to modify the schedule. The schedule page looks like:



To add a new schedule, click the [Insert] button. To modify existing schedules, check the schedules and click the [Modify] button. To delete existing schedules, check the schedules to be deleted and click the [Delete] button. The insert and modify page looks like:



Customize the Type, Interval, Base, and Action to desirable settings and click [Modify] button to confirm the change.

Leave the Session attribute empty to use a default browser. Give a special Session name if you want the robot to launch a special browser for the scheduled task.

6. Frequently Asked Questions

6.1. Can I create a dummy action as a placeholder?

Yes. You can add a Schedule action, without any associated schedule items, as a dummy action. You can also associate events with this dummy action.

6.2. The robot skips certain actions during navigation!

This most probably is because of the slow Internet connection, where the robot has determined a timeout for the previous action, and continues on the next action. To solve this problem, you can set a slower robot navigation speed from the menu: Settings → Navigation Speed → Very Slow; or you manually introduce a sleep after the previous action (add an event "After each tuple", and set the value as "sleep(10)", which sleeps for 10 seconds).

If the previous action is a Click action, it may because the action was setup not to wait for the target page (this is useful when clicking on some Ajax actions). You can right-click on the action and select the "Property …" from the popup menu to see if it waits for navigation. Change to "Wait" if it is "No wait".

6.3. How can I scrape data from pop-up windows?

You can either open pop-up windows in the current browser (using menu: Settings → Popup → Open in Current Browser) or check the "To open in new browser" in the action property.

6.4. Save Variables does not work with MS-Access database, but it works with XML

Your table names or field names in MS-Access database may include special characters, such as spaces or non alpha-numerical characters.

7. Additional Information

Online Forum (preferred): http://irobotsoft.org/bb/
Technical Support: support@irobotsoft.com
Donation: http://irobotsoft.com/donate.html
Collaboration: info@irobotsoft.com

Copyright

IrobotSoft.com May. 2005
All rights reserved.